Unsupervised Machine Learning Paradigms for the Representation of Music Similarity and Structure.

Axel Marmoret

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par
Axel MARMORET

Unsupervised Machine Learning Paradigms for the Representation of Music Similarity and Structure.

Paradigmes d'Apprentissage Automatique Non-Supervisés pour les Représentations de la Similarité et de la Structure Musicale.

Thèse présentée et soutenue à l'Université de Rennes 1, IRISA, le 2 Décembre 2022
Unité de recherche : IRISA (UMR CNRS 6074)

Rapporteurs avant soutenance :

Cédric FÉVOTTE	Directeur de Recherche à l'Institut de Recherche en Informatique de Toulouse (IRIT), Université de Toulouse
Mathieu GIRAUD	Directeur de Recherche au CRIStAL (UMR CNRS 9189), Université de Lille

Composition du Jury :

Présidente :	Elaine CHEW	Professeure au King's College London
Examinateurs :	Cédric FÉVOTTE	Directeur de Recherche à l'Institut de Recherche en Informatique de Toulouse (IRIT) Université de Toulouse
	Mathieu GIRAUD	Directeur de Recherche au CRIStAL (UMR CNRS 9189), Université de Lille
	Aline ROUMY	Directrice de Recherche à l'IRISA (UMR CNRS 6074), Inria, Université de Rennes 1
	Elaine CHEW	Professeure au King's College London
Dir. de thèse :	Frédéric BIMBOT	Directeur de Recherche à l'IRISA (UMR CNRS 6074), Université de Rennes 1
Co-enc. de thèse :	Jérémy COHEN	Chargé de Recherche au CREATIS, CNRS, Université de Lyon

Invité(s) :

Nancy BERTIN	Chargée de Recherche à l'IRISA (UMR CNRS 6074), Université de Rennes 1
Simon LEGLAIVE	Maître de Conférences à CentraleSupélec, IETR (UMR CNRS 6164)

“...”

L'âne Rouge - L'effondras

Résumé

Traiter informatiquement la musique ? La musique fait partie des formes d'art les plus répandues sur la planète, avec des origines remontant à 10 000 ans [Law88], si ce n'est plus. Néanmoins, la musique est également un objet complexe, en particulier sous sa forme audio. Alors que le cerveau humain produit inconsciemment un travail d'analyse de la musique lors de l'écoute, même sans entraînement ni pratique musicale (reconnaissance des instruments présents dans un morceau, perception de l'organisation du morceau, détermination de la pulsation pour taper dans ses mains, ...), l'analyse musicale par traitement informatique s'avère être une tâche délicate.

Un domaine de recherche est dédié à ce sujet : la Recherche d'Information Musicale (MIR). Ce domaine est aujourd'hui en plein essor, notamment grâce au déploiement de gigantesques bases de données et de services d'écoutes (plateformes de "streaming" notamment) alliant la possibilité de traiter de grandes masses de données avec le besoin de plus en plus étendu de traitement automatisés.

Les tâches du domaine du MIR sont nombreuses. Citons par exemple la transcription automatique (représenter un extrait audio par sa partition), la séparation de sources (séparer les parties audios de chacun des instruments) ou encore la génération automatique de musique. Le MIR se trouve au centre de nombreux domaines scientifiques, tels que les mathématiques, l'informatique, l'acoustique, la musicologie, la biologie humaine, l'histoire, ... Ce vaste domaine ne saurait être parfaitement résumé en quelques lignes.

Les techniques récentes de traitement automatique de la musique peuvent s'appuyer directement sur le signal audio (typiquement, les architectures récentes de réseaux de neurones, tels [DS14]), mais la majorité des travaux dans le domaine du MIR s'appuient sur des **descripteurs temps-fréquence** de la musique, le plus connu étant la Transformée de Fourier à Court Terme, également appelés **spectrogrammes**.

Quel est le périmètre de cette thèse ? Au sein du domaine du MIR, cette thèse concerne plus particulièrement la tâche de **segmentation structurelle** de la musique, c'est-à-dire la tâche consistant à fournir une organisation simplifiée de la musique en sections (couplet, refrain, solo, ...), et, plus particulièrement, identifier les frontières entre

ces différentes sections.

Cette tâche, déjà traitée par de nombreux chercheurs (voir les synthèses de Paulus *et al.* [PMK10] ou de Nieto *et al.* [Nie+20]), n'est pas considérée comme résolue : les systèmes automatiques n'étant pas suffisamment performants à ce jour. D'autant plus que la segmentation musicale manque d'une définition concise et précise, car la structure musicale est en partie subjective et parfois ambiguë. Cette thèse contient une présentation détaillée des algorithmes de segmentation structurelle existants.

Comment déterminer automatiquement la segmentation structurelle ? La segmentation structurelle, *i.e.* déterminer les frontières entre différentes sections musicales, peut s'effectuer de différentes façons [PMK10; Nie+20]. En général, les algorithmes existants se concentrent sur 4 critères : l'**homogénéité**, la **nouveauté**, la **répétition**, et la **régularité**.

Le critère d'homogénéité suppose que les différents éléments musicaux (notes, accords, tonalité, timbres présents, ...) constituant une section musicale devraient être similaires les uns avec les autres, et se concentre ainsi sur le fait d'identifier des frontières entre des blocs homogènes. La nouveauté est le pendant de l'homogénéité : ce critère fait l'hypothèse qu'un contraste très marqué entre deux éléments musicaux devrait constituer une frontière. En particulier, une forte nouveauté n'est possible que lorsqu'elle est précédée par un élément très homogène (une rupture d'homogénéité), et inversement.

Le troisième critère, la répétition, repose sur l'idée qu'une section n'est pas définie de façon isolée, mais par la répétition d'une séquence d'éléments, par exemple une ligne mélodique. Pour illustrer ce principe, considérons le refrain dans une chanson de type "Pop". Le refrain est souvent considéré "refrain" car les paroles sont consistantes entre les différentes répétitions, de même que la mélodie de la voix et les lignes instrumentales. Prise isolément, ces lignes peuvent être peu homogènes, mais leur répétition à différents endroits du morceau suggère une cohérence et une appartenance à un même ensemble. Finalement, le critère de régularité se base sur l'idée que les segments dans une même chanson (voir dans un même genre) doivent être de même taille.

Les algorithmes de segmentation structurelle existants s'appuient en général sur un outil appelé "matrice d'autosimilarité", présenté dans la Figure 1. Cette matrice représente la similarité (au sens large) entre toutes les paires d'instants temporels d'un morceau. Les parties très homogènes apparaissent alors comme des blocs rectangulaires ou carrés dans la matrice (représentés en gris dans le schéma de la Figure 1), la nouveauté se conçoit

comme le point de contact entre deux blocs, et la répétition est représentée par des portions de sous-diagonales (parallèles à la diagonale principale). Notons que la diagonale elle-même représente la similarité entre un instant temporel et lui-même, et est donc peu informative. Les coefficients de la matrice d'autosimilarité sont en général calculés comme le produit scalaire normalisé entre les vecteurs de descripteurs correspondant à deux instants temporels.

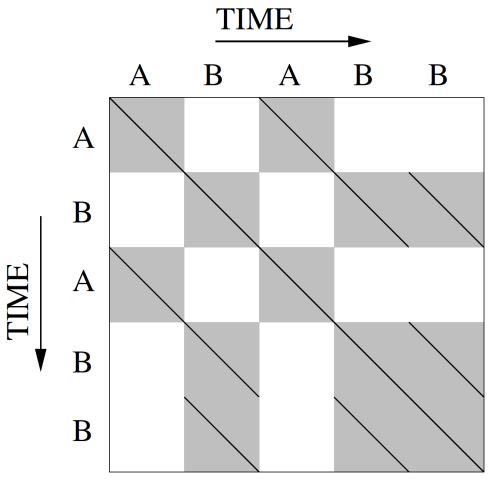


Figure 1 – Un exemple schématique et idéalisé de matrice d'autosimilarite, extrait de [PMK10].

Bien que la matrice d'autosimilarité soit parfois calculée directement sur la représentation en descripteurs de la musique (citons en particulier les travaux de Foote [Foo00]), les avancées récentes dans le domaine s'attèlent à retravailler cette représentation en descripteurs afin de plus clairement faire apparaître la structure dans l'autosimilarité (par exemple [NJ13; ME14a; SNB21; Wan+21], cette liste n'étant pas exhaustive).

Comment se situe cette thèse dans la littérature scientifique ? Tout d'abord, il faut noter que cette thèse se concentre sur l'analyse de la musique "Pop". Bien que, conceptuellement, les travaux présentés pourraient être utilisés pour d'autres genres musicaux, la musique "Pop" est particulièrement régulière dans sa structure, ce qui permet de traiter dans l'immédiat un problème plus simple que le problème de segmentation "global", *i.e.* indépendant du genre musical.

Les travaux de cette thèse se concentrent sur trois axes principaux : le traitement de la musique par mesures, l'utilisation d'un nouvel algorithme de segmentation (CBM), et l'utilisation de méthodes de compression de l'information musicale. Ces parties sont explicitées ci-dessous.

1. Un traitement de la musique par mesures musicales. Nous considérons dans cette thèse que l'échelle temporelle de la mesure est l'échelle la plus pertinente pour analyser la structure musicale, en particulier pour la musique "Pop". En effet, nous supposons que les frontières structurelles apparaissent presque exclusivement entre deux mesures. De plus, nous considérons que les phrases et motifs musicaux se développent à cette échelle, nos travaux visant ensuite à comparer les différentes mesures pour définir et identifier les segments selon ces motifs.

   Ce traitement par mesures concerne les données initiales : à partir du spectrogramme d'une chanson (représentation en descripteurs temps-fréquence de base), et en estimant les débuts de chaque mesure (à l'aide d'une librairie extérieure), notre travail consiste à séparer ce spectrogramme en une collection de spectrogrammes, chacun se limitant à une unique mesure de la chanson. Le processus est représenté en Figure 2.

   Il est à souligner que l'étape d'estimation des mesures est issue de travaux extérieurs et, bien qu'importante, n'a pas fait l'objet d'un travail de recherche dans cette thèse.

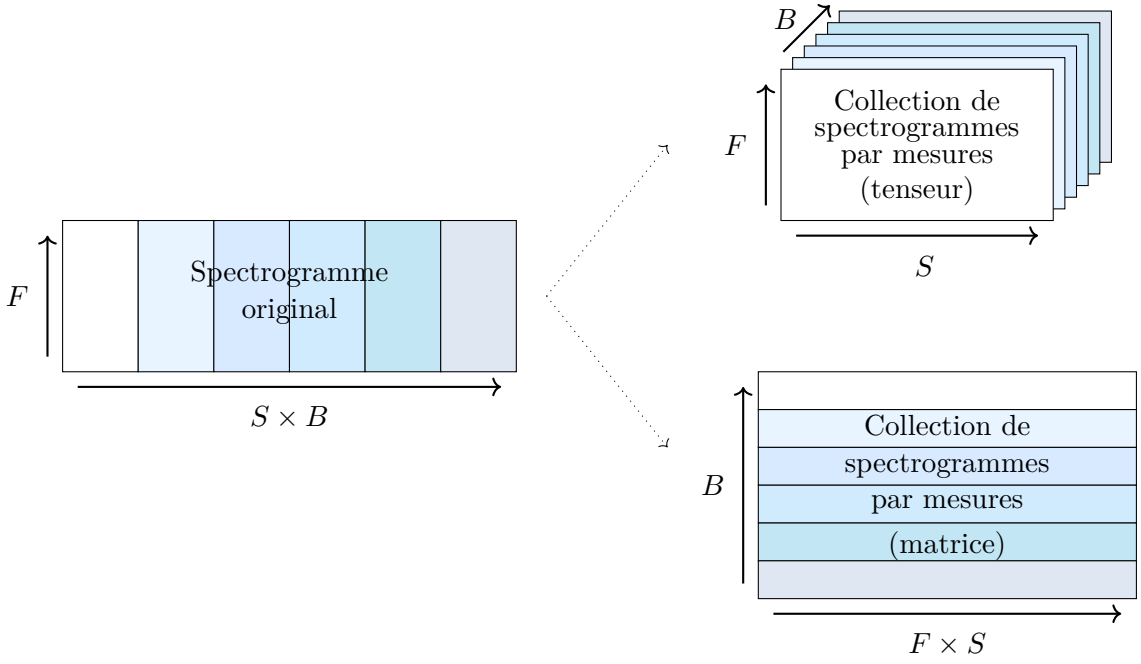


Figure 2 – Traitement par mesures de la musique, en tenseur et en matrice. Les dimensions F, S, B sont représentées afin de faciliter la compréhension du processus.

2. Le développement d'un nouvel algorithme de segmentation, appelé algorithme "CBM" pour *Convolutive Block Matching*. Cet algorithme est un algorithme de

programmation dynamique, *i.e.* un algorithme qui divise une tâche initiale (ici, la tâche de segmentation structurelle) en un ensemble de sous-tâches (ici, la détermination d'un score pour chaque potentiel segment) qui, résolues récursivement et comparées au sein d'un problème d'optimisation, fournissent une solution globale à la tâche initiale.

Ainsi, la tâche de segmentation structurelle se trouve ici réduite à la définition d'un score pour chaque segment, et la programmation dynamique trouve ensuite une solution optimale, maximisant la somme de ces scores.

Dans cette thèse, l'algorithme CBM se concentre sur les principes d'homogénéité-nouveauté et de régularité, mais de futurs travaux pourraient également incorporer le principe de répétition. L'algorithme CBM est détaillé dans le Chapitre 3.

3. L'utilisation de nouvelles méthodes de compression pour analyser la musique en mesures. En particulier, cette thèse étudie la décomposition nonnégative de Tucker (NTD), un modèle de factorisation tensorielle représenté en Figure 3 et détaillé dans le Chapitre 4, ainsi que des techniques de compression matricielles plus classiques que sont la factorisation en matrices nonnégatives (NMF), l'analyse en composantes principales (PCA) et les autoencodeurs, détaillées dans le Chapitre 5.

   En particulier, les autoencodeurs utilisés dans cette thèse n'ont pas pour objectif l'apprentissage de représentations généralisées entre différentes chansons, mais sont utilisés comme méthodes de compression à l'échelle de chaque chanson, ce qui, à notre connaissance, est un nouveau paradigme.

   Enfin, cette thèse explore une combinaison entre les paradigmes NTD et autoencodeurs, les deux techniques présentant des similarités conceptuelles et mathématiques, qui sont détaillées dans le Chapitre 6. Ce paradigme, dénommé "AE-NTD" est, à notre connaissance, le premier travail combinant ces deux techniques de compression.

Les articulations de cette thèse sont schématisées dans la Figure 4.

Quelles sont les conclusions expérimentales de ces travaux ? L'étude de ces méthodes s'accompagne d'expérimentations visant à évaluer leur potentiel quant à l'étude de la structure musicale et, dans un sens plus large, quant à l'analyse musicale, via la découverte de motifs à l'échelle de la mesure. Les conclusions sont plurielles :

— L'algorithme CBM, dans sa version actuelle et utilisant la similarité cosine (similarité standard), permet d'obtenir des performances proches de celles obtenues avec

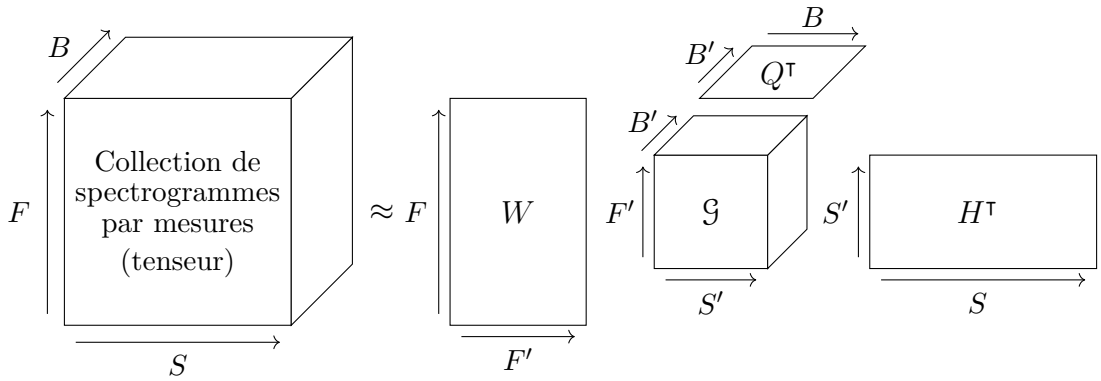


Figure 3 – Décomposition nonnégative de Tucker (NTD) du tenseur construit à partir de la collection de spectrogramme par mesures. La NTD résulte en trois matrices de facteurs W, H, Q, et un tenseur $\mathcal{G}$ (appelé "cœur").

les algorithmes non supervisés état-de-l'art, typiquement [Foo00; NJ13; Ser+14; ME14a]. Il n'est en revanche pas compétitif avec l'algorithme état-de-l'art global (avec supervision) [GS15b], lequel requiert toutefois de grandes bases de données étiquetées manuellement.

— L'augmentation du contraste entre zones très similaires (donc très homogènes) et peu similaires dans la matrice d'autosimilarité améliore les performances de l'algorithme CBM. Cela s'explique principalement par le fait que les zones très homogènes sont identifiées en tant que segments par l'algorithme, et ainsi, un haut contraste permet de désambiguïser les frontières entre les blocs.

Dans cette thèse, deux stratégies se sont avérées pertinentes pour augmenter ce contraste (et donc les performances de l'algorithme CBM) :

— L'utilisation de fonctions de similarité différentes de la similarité cosine, en particulier l'utilisation d'un noyau gaussien, fonction non-linéaire.

— L'utilisation de méthodes de compression pour représenter les différentes mesures de la chanson. En effet, compresser les différentes mesures permet de ne conserver que certains traits distinctifs du contenu musical, permettant ainsi de mieux distinguer les similarités et les dissimilarités.

Dans ces deux conditions, l'algorithme CBM est compétitif avec l'état-de-l'art global [GS15b], alors qu'il ne nécessite aucune phase d'apprentissage.

En revanche, l'utilisation de ces deux techniques simultanément (*i.e.* utiliser des fonctions de similarité autres que la similarité cosine sur des représentations compressées) ne permet pas d'augmenter les performances par rapport à l'une ou l'autre utilisée seule.

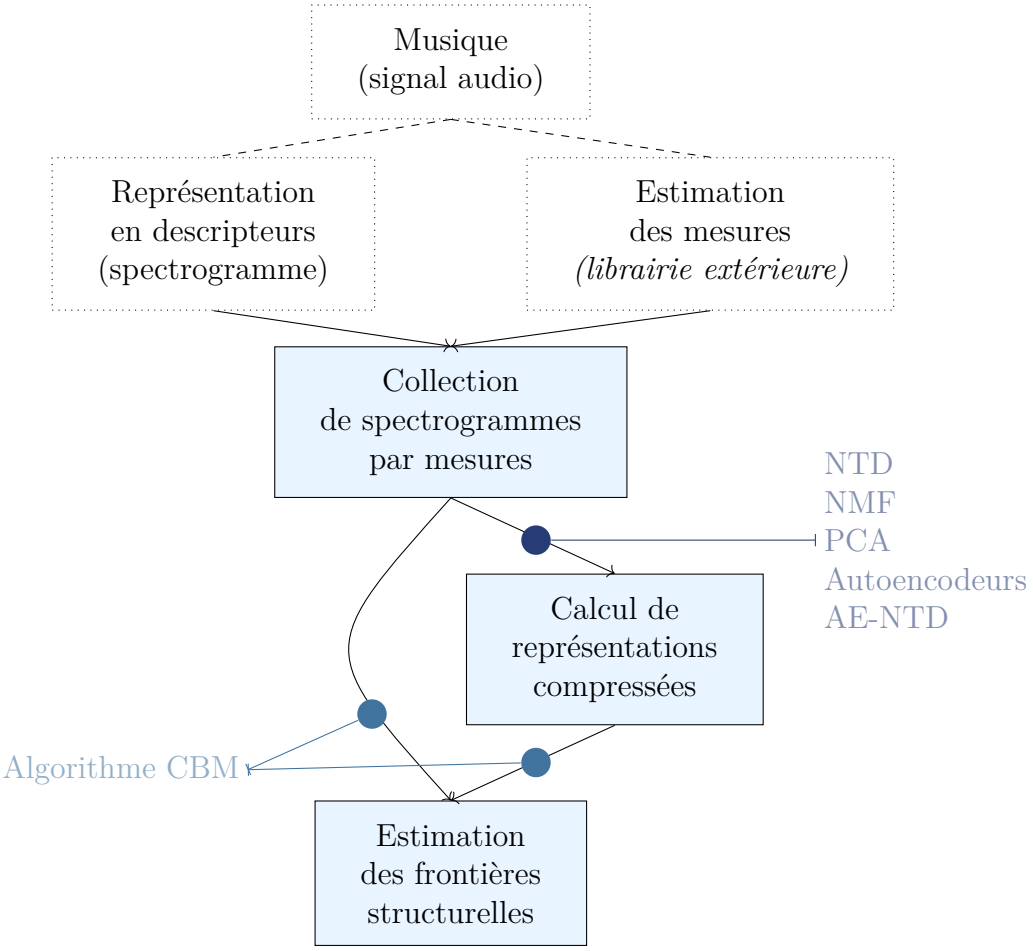


Figure 4 – Articulations de la thèse. Cette thèse présente les étapes de calcul de la collection de spectrogrammes en mesures, l'algorithme CBM et les différentes méthodes de compression. *Les étapes de représentation en descripteurs et d'estimation des mesures sont issues de travaux extérieurs et, bien qu'importantes, n'ont pas fait l'objet d'un travail de recherche dans cette thèse.*

Trois logiciels libres ont été développés durant ce doctorat, chacun étant dédiée à une partie du travail : l'algorithme CBM est inclus dans la librairie *as_seg* [MCB22a], les algorithmes de résolution de la NTD et de la NMF sont inclus dans la librairie *nn_fac* [MC20], et l'ensemble du code informatique associé aux représentations compressées est rassemblé dans la librairie *BarMusComp* [MCB22b].

— Certaines techniques de compression, en particulier la NTD, permettent de représenter les différentes mesures selon des motifs, eux-mêmes obtenus dans la factorisation. Pour la NTD, cela est rendu possible grâce aux contraintes de nonnégativité et à la structure de la factorisation.

Ainsi, les différents facteurs issus de la NTD permettent de définir des "patterns

musicaux", motifs à l'échelle d'une mesure, représentant une partie du contenu musical original. Ces patterns pourraient incidemment être utilisés pour de la recomposition musicale [SKG19] ou de l'analyse.

— Dans le cadre des méthodes de compression, un compromis apparaît entre les performances de segmentation et l'interprétabilité des résultats. En effet, alors que la NTD permet d'interpréter le contenu de la factorisation, les techniques de compression non contraintes (ici, PCA et autoencodeurs) n'ont pas cette propriété, mais obtiennent de meilleures performances dans la tâche de segmentation structurelle. Ce compromis est particulièrement clair dans l'étude de l'AE-NTD, où les meilleures performances en termes de segmentation sont obtenues sans contrainte, mais où l'interprétabilité des résultats s'améliore avec l'ajout d'information a priori liée à la NTD.

Quelles ouvertures à cette thèse ? Tout d'abord, les conclusions expérimentales se limitent pour l'instant au corpus étudié et nécessitent d'être confirmées sur d'autres données, notamment de la musique appartenant à d'autres styles que la Pop. L'impact de l'étude de la musique par mesures mérite également d'être approfondi par l'étude de genres où la division en mesures est plus ambiguë ou moins régulière (par exemple avec de nombreux changements de tempo).

Ensuite, l'algorithme CBM peut être amélioré et étendu dans différentes directions, permettant notamment de favoriser le principe de répétition, ou bien de mitiger entre les différents critères. Certaines pistes existent déjà, en particulier en utilisant les "noyaux" développés par Shiu *et al.* [SJK06] dans l'algorithme CBM.

De la même manière, les techniques de compression prouvent leur efficacité mais nécessitent d'être étudiées plus en détail afin de mieux capturer la structure musicale. Chaque technique ouvre des pistes potentielles spécifiques, qui ne sauraient être résumées ici, et sont donc à retrouver dans les conclusions des chapitres associés. Toutefois, de manière générale, il semblerait pertinent d'ajouter une phase de supervision (ou de semi-supervision) aux différentes techniques étudiées dans cette thèse lorsque possible.

Les techniques dites "interprétables" (en particulier la NTD et l'AE-NTD) mériteraient d'être approfondies, permettant d'étendre les conclusions à d'autres tâches que la recherche de structure musicale.

Remerciements

Quelle aventure !

Je ne saurais remercier en quelques lignes toutes les personnes qui ont rendu ce doctorat possible, de près ou de loin, scientifiquement ou moralement, en présentiel ou en distanciel (maintenant que la maladie Dont-On-Ne-Doit-Pas-Prononcer-Le-Nom a rendu le distanciel inévitable). Permettez-moi tout de même d'essayer, en commençant par m'excuser auprès de celleux que j'aurais malheureusement oublié·e·s !

Tout d'abord, je remercie infiniment mes 2 encadrants : mon directeur de thèse **Frédéric Bimbot** et mon directeur des travaux **Jérémy Cohen**. **Frédéric**, tu auras su m'apporter toute ton expérience, tes bonnes idées et tes qualités d'écriture, accompagné d'une sérénité toujours bienvenue pour envisager le déroulé de la thèse et me permettre de me décider dans mes idées (nombreuses et hétérogènes, pour ne pas dire loufoques). Ton point de vue aura été très important pour prendre de la hauteur sur les travaux réalisés, ainsi que pour apporter une cohérence globale à tout ce qui a été réalisé durant ces 3 ans. De manière générale, tu auras toujours su maintenir le cap de la thèse et diriger nos idées dans la bonne direction ! **Jérémy**, tu auras été mon *partner-in-crime* durant ces 3 ans (et demi en comptant le stage pre-thèse !). Malgré nos longs débats sur ce que signifie le Machine Learning, l'apprentissage, et la musique de manière générale, tu auras su être très pédagogique et patient pour me permettre d'apprendre énormément, en particulier sur les méthodes d'optimisation/de factorisations de données, qui constituent le coeur de ces travaux. Jamais nous n'aurions pu mener à bien tous ces travaux sans tes connaissances et ton point de vue, toujours éclairant et pertinent (je sais, j'aurais dû plus t'écouter... Ca fait partie de l'apprentissage !). Ta gentillesse, ton écoute et ton soutien auront également été déterminants dans mes moments de doute et de faiblesse, car un doctorat est une aventure humaine avant tout ! Il n'aura pas toujours été facile de nous comprendre tous les trois, mais nous avons toujours su discuter avec l'envie de réussir ensemble, et c'est ce que nous avons fait ! Ces travaux portent notre marque à chacun, et je suis extrêmement fier du résultat, merci à vous !

J'aimerais également remercier **Nancy Bertin**, mon encadrante officieuse pour environ la moitié de cette thèse. Bien que n'étant pas "officiellement" dans la direction, tu

as su m'apporter tes connaissances et ton regard expert sur la musique, notamment ses propriétés acoustiques et techniques, et sur tout le domaine MIR en général. Les épreuves de la vie ont fait que nous n'avons pas pu collaborer sur la totalité des 3 années, mais ton impact a tout de même été déterminant dans la réalisation de ces travaux, et tu es et resteras une source d'inspiration énorme à mes yeux ! De la même manière, merci à **Simon Leglaive** qui a beaucoup participé aux travaux de fin de thèse, et apporté beaucoup de connaissances (notamment sur les réseaux de neurones et les avancées récentes en audio), ainsi qu'un point de vue toujours pertinent, intéressant et rafraîchissant. A mon grand regret, je n'ai pas pris le temps de creuser toutes les pistes que tu nous as proposé, malgré les promesses qu'elles apportaient. J'espère être amené à collaborer avec vous de nouveau dans le futur !

Je remercie l'ensemble des membres du jury, **Elaine Chew**, **Cédric Févotte**, **Mathieu Giraud** et **Aline Roumy**, de m'avoir fait l'honneur de juger ce travail de thèse ainsi que pour la discussion passionnante durant la soutenance. Merci tout particulièrement à **Cédric** et **Mathieu** d'avoir accepté de relire et de juger ce manuscrit dans le détail. Je remercie également **Emmanuel Vincent** et **Aline** de nouveau pour leur participation à mon comité de suivi, leur écoute, leur investissement et leurs remarques très pertinentes tout au long de cette thèse.

Cette thèse aura été l'occasion de nombreuses rencontres, au premier lieu desquels mes nombreux collègues: **Stéphanie Gosselin-Lemaile** tout d'abord, qui a largement contribué à rendre cette thèse plus facile ; les permanents, contractuels, docteurs et ex-doctorants (désormais docteurs !) de l'équipe PANAMA: **Kilian**, **Diego**, **Antoine**, **Clément G.**, **Cassio**, **Valentin**, **Rémi G.**, **Clément E.**, **Ewen**, **Romain**, **Mohammed**, mais aussi du labo CREATIS à Lyon: **Nico-Raoul**, **Lulu**, **Franck**, **Louise**, **Pierre**, **Sophie**, **Mom**, **Juliette**, **Matthis** et beaucoup d'autres. Vous avez tous·te·s été de super rencontres !

Comment ne pas citer mon super collègue/pas collègue et co-bureau **Stéphane Kastenbaum**, qui aura été mon plus fidèle partenaire durant toute la rédaction de ce manuscrit, et que j'ai pu déranger dans sa propre rédaction par mes nombreuses remarques très peu intéressantes et autres débats infinis pour finalement avoir les mêmes idées. Merci à toi, tu es un super ami et je suis content d'avoir pu partager ce temps en ta compagnie ! Merci également aux copaines **Lucas** (TGVMax king), **Rémi C.** (alias Cat), **Jean-Joseph/JJ**, **Louise**, **Elodie**, **Thomas C.**, **Salomé**, **Katja**, **Guéno**, **Maxime**, **Adèle**, **Thomas R.** et tous les copaines de ce cercle, en particulier **Marie** (ma super "pote"), c'était très

chouette de partager tout ce temps à l'IRISA (et en dehors) en votre compagnie !

Avant cette thèse, il y avait le Master SIF, que de chemin parcouru **Cédric** et **Samuel** ! J'ai une pensée toute particulière pour le 3ème confinement passé en autarcie dans l'Anjou **Cédric** (et Ulysse le poney), qui restera surement un des moments les plus fantastiques de cette thèse.

Merci à mes potos de musique, **Luc** en premier lieu. On aura galéré mais on aura réussi à faire de la musique vachement trop bien avec NDC, également grâce à **Mathéo**, t'as su t'approprier les chansons pour un rendu génial ! Merci aussi à **Léo**, **Sacha**, **Arthur D.**, **Arthur N.**, **Yoann** pour la super musique que vous faîtes, je me sens honoré de vous avoir rencontré !

Merci à tous mes supers meilleur·e·s copaines qui sont proches depuis bieeen longtemps désormais, **Vincent** (19 ans qu'on se connaît sur 25, pas mal !), **TT** (alias Frangin, t'as probablement été mon plus grand soutien), **Hugo**, **Fanny**, **Zou**, **Carmen**, **Pierre**, **Aurélie**, **Thomas** (alias Panda), **Marion**, **Kallan**, **Coco**, **Solène**, toute la Timmy Carnac (**Didi**, **Clotteau**, **Nico**, **Paul**, **Seb**, **Antoine**, **Diane**, **Mathieu**), **Axel** (l'autre, pas moi voyons), **Jacquo**, **Orianne**, **Gaël**, **Chauty**. Votre soutien a été incroyable durant toutes ces années, et qu'est-ce qu'on se marre ensemble ! J'ai de la chance de pouvoir compter sur vous tous, du fond du coeur ! De plus, merci **Océane R.** pour les discussions et réflexions passionnantes, qui m'accompagnent encore.

Merci **Michel**, mon super prof de batterie, tu es responsable de mon amour et de ma passion pour cet instrument. Je n'aurais pu rêver meilleur prof, toi qui à une telle joie de transmettre ton art, et qui m'a autant fait rire que fasciné par ton jeu. Merci, merci, merci !

Enfin, un merci énorme pour ma famille, qui m'a toujours soutenu, encouragé et motivé dans mes études (et autres envies/projets, surtout musicaux !), merci **Papa**, merci **Maman**, merci **Grande Tos** (et désolé pour le bruit causé par la batterie depuis toutes ces années...) ! Merci **Mamie**, **Tatie**, **Laurent**, **Anaïs**, **Théo**, et merci **Grand-père**, merci **Grand-mère** ! Je finis par vous, **GP/GM**, car vous nous avez non seulement toujours soutenus et poussés, Marine et moi, mais vous nous avez surtout transmis énormément de connaissances par votre temps passé à nos côtés dans notre jeunesse. Merci **Grand-père** pour m'avoir transmis ton amour des maths, cette thèse n'est évidemment pas une thèse de maths, mais je n'aurais jamais pu aimer réaliser des décompositions tensorielles ou apprendre n'importe quelle méthode de Machine Learning sans ton soin à m'apprendre le domaine tout jeune !

En dernier lieu, j'aimerais remercier toutes ces personnes qui m'accompagnent au quotidien sans le savoir, qui me font ressentir énormément de sensations et sentiments grâce à leur art, et qui rythment ma vie et mes humeurs. Je parle des musiciens de **TOOL**, de **Rage Against The Machine**, de **All Them Witches**, de **Stoned Jesus**, du génie **Jimi Hendrix**, de **L'Effondras**, de **Primus**, de **Lamb of God**, de **Gojira**, de **Gong**, de **Led Zeppelin**, de **Pink Floyd**, et de bien d'autres, ne pouvant pas réaliser de liste exhaustive. Merci à vous, infiniment, cette thèse étant avant tout une tentative de ma part de vivre de la musique et de vous rendre hommage.

Table of Contents

1 Introduction **21**

I Music Structural Segmentation **27**

2 Music Processing and Music Structure Analysis **29**

2.1 Introduction 30

2.2 Music Processing (generalities) 30

2.2.1 Audio and Symbolic Representations of Music 30

2.2.2 Time in Music 33

2.2.3 Digital Signal Processing 36

2.2.4 Feature Representations 37

2.3 Music Structure Analysis 44

2.3.1 Boundary Retrieval - Structural Segmentation 46

2.3.2 Labelling of Segments 53

2.3.3 Feature Representation in MSA 54

2.3.4 Metrics and Datasets 54

2.4 Focus on Barwise Music Processing 58

2.4.1 Motivations 58

2.4.2 Barwise Processing of Inputs 60

2.4.3 Dataset Analysis 63

2.5 Conclusions 64

3 Convolutive "Block-Matching" Segmentation Algorithm **65**

3.1 Introduction 66

3.2 Autosimilarity Matrix 67

3.2.1 Cosine Autosimilarity Matrix 67

3.2.2 Covariance Autosimilarity Matrix 68

3.2.3 RBF Autosimilarity Matrix 68

3.3 Convolutive "Block-Matching" Algorithm 69

3.3.1 Structural Segmentation Solved as a Dynamic Programming Algorithm . . . 70
3.3.2 Convolution Kernels . . . 75
3.3.3 Penalty Functions . . . 79
3.4 Experiments . . . 81
3.4.1 Autosimilarity Matrices . . . 82
3.4.2 Convolution Kernels . . . 84
3.4.3 Penalty Functions . . . 86
3.4.4 Experimental Conclusions . . . 87
3.5 Conclusions . . . 88

II Barwise Compression Methods 91

4 Nonnegative Tucker Decomposition - NTD 93
4.1 Introduction . . . 94
4.2 Elements of Tensor Algebra . . . 95
4.2.1 Tensors . . . 95
4.2.2 Nonnegative Tucker Decomposition Model . . . 98
4.2.3 Nonnegative Matrix Factorization - NMF . . . 99
4.3 Algorithms for NTD . . . 100
4.3.1 Alternating Scheme . . . 100
4.3.2 Euclidean-NTD: Optimizing the Euclidean Distance . . . 101
4.3.3 β-NTD: Optimizing the β-divergences . . . 107
4.4 Musical Barwise Interpretation . . . 110
4.4.1 TFB Tensor . . . 111
4.4.2 Factors Interpretation . . . 112
4.5 Experiments . . . 119
4.5.1 NTD for Structural Segmentation . . . 119
4.5.2 NTD for Pattern Uncovering . . . 127
4.6 Conclusions . . . 135

5 Linear and Nonlinear Barwise Compression Schemes 137
5.1 Introduction . . . 138
5.2 Barwise TF matrix (reminder) . . . 139

5.3 Low-Rank Factorizations . . . 140
5.3.1 Barwise NMF . . . 140
5.3.2 Principal Component Analysis - PCA . . . 141
5.3.3 Structural Segmentation Experiments . . . 142
5.4 Single-Song AutoEncoders . . . 149
5.4.1 Motivations . . . 150
5.4.2 Paradigm Description . . . 150
5.4.3 Network Architecture . . . 151
5.4.4 Structural Segmentation Experiments . . . 156
5.4.5 Additional Experiments: Extending the SSAEs . . . 162
5.5 Conclusions . . . 169

6 AE-NTD: Single-Song AutoEncoders Based on NTD 173
6.1 Introduction . . . 174
6.2 Mathematical Formalism . . . 175
6.2.1 Reminder of NTD at the Barscale . . . 175
6.2.2 Reminder of SSAE . . . 175
6.2.3 AE-NMF . . . 176
6.2.4 AE-NTD . . . 177
6.3 Motivations . . . 180
6.3.1 Study the Performance-Interpretability Trade-Off . . . 180
6.3.2 Increasing the Expressiveness of the Decomposition . . . 181
6.3.3 Technical Benefits . . . 181
6.4 Experiments . . . 182
6.4.1 Technical Details . . . 182
6.4.2 Interpretation of the NTD-based Decoder . . . 182
6.4.3 Structural Segmentation . . . 185
6.4.4 Pattern Uncovering . . . 190
6.4.5 Experimental Conclusions . . . 192
6.5 Conclusions . . . 193

7 Conclusion 195

List of Figures 199

List of Tables **207**

Bibliography **208**

Appendices 229

A Experimental Details **229**
A.1 Features, in Details . . . 229
A.1.1 STFT . . . 229
A.1.2 Mel Spectrograms . . . 229
A.1.3 Chromagram . . . 230
A.1.4 MFCC . . . 230
A.2 Dataset in Learning/Test Paradigm . . . 230

B Technical Choices for the SSAE **233**
B.1 FC SSAE . . . 233
B.2 Batch Size . . . 234
B.3 Batch Normalization . . . 235

C General Init for the KL-AE-NTD **241**

D Additional Articles, Not Presented in this Thesis **245**
D.1 Semi-Supervised Convolutive NMF for Automatic Piano Transcription . . . 245
D.2 Polytopic Analysis of Music . . . 254

Chapter 1

Introduction

Computational analysis of music? Music is an artistic, human form of expression, which may trace back to 10,000 years [Law88] or even more. Still, music is a complex object, in particular in its audio form. While the human brain unconsciously analyses the musical content while listening, even without training or musical practice (recognizing the instruments in a song, understanding the organization of a song, finding the pulse to clap hands, ...), computational musical analysis is a challenging topic.

The Music Information Retrieval (MIR) research field is dedicated to this topic. This field is growing nowadays, notably due to the presence of large datasets, which combines the need for automated processing of musical information with the possibility to study gigantic quantities of data, which is important in recent supervised learning schemes.

The MIR field covers numerous tasks, for instance automatic music transcription [Ber09] (computing the musical score from an audio excerpt), source separation [OF09] (retrieving the original audio contribution of each source in an audio mixture), or automatic music generation [BHP20]. It is therefore at the crossroad of numerous scientific domains, *e.g.* mathematics, computer science, acoustics, musicology, human biology, history, ...

A common platform for comparing the algorithms relative to the different MIR subtasks has been developed in the last two decades, called "Music Information Retrieval Evaluation eXchange" [1] (MIREX) [Dow08], allowing to compare algorithms on standard benchmarks.

What is the scope of this thesis? This thesis studies the Music Structure Analysis task [PMK10; Nie+20], which consists of representing a song in sections (such as "chorus", "verse", "solo" etc). It can be seen as the retrieval of a simplified organization of the song, at the macroscopic scale, but lacks a clear and concise definition. In particular, this thesis studies the subtask of **structural segmentation**, which consists of retrieving the boundaries between sections, *i.e.* segmenting the song in non-overlapping segments.

1. https://www.music-ir.org/mirex/wiki/MIREX_HOME

How to identify structural segmentation? Structural segmentation can be algorithmically achieved in numerous ways, as presented in Section 2.3 of this thesis. Firstly, it should be noted that, while some recent works in MIR study the raw waveform (typically "end-to-end" neural networks, such as [DS14]), the vast majority of works use features to describe the musical content. The particular feature descriptions used in this thesis are different forms of **spectrograms**.

Existing algorithms for the estimation of structural segmentation mainly focus on 4 criteria: **homogeneity**, **novelty**, **repetition**, and **regularity**. The homogeneity criterion assumes that musical elements (notes, chords, tonality, timbre, ...) should be similar to constitute a section. Novelty is the counterpart of homogeneity: this criterion considers that boundaries must be placed between consecutive musical elements that are highly contrastive. A high contrast, *i.e.* a high novelty, is only possible between two homogeneous zones ("break" of homogeneity), and, conversely, homogeneity is evaluated relatively to the dissimilar portions in the song.

The third criterion, repetition, does not consider segments locally, but rather relies on a global approach of the song. The rationale is that segments are constituted of recurring motifs, for instance a melodic line, and these motifs may be heterogenous. This criterion may be illustrated by the chorus in a song, which is generally defined as the most consistent section across the song, both in terms of lyrics and instrumental lines: a chorus, on its own, may be heterogenous and contrasting, but its repetition suggests that this is indeed a section. Finally, the regularity criterion assumes that, within a song (and even within a same musical genre), segments should be of similar sizes.

What are the contributions of this thesis? First of all, this thesis focuses on "Pop" music. Even if, conceptually, the methods presented could be used for other genres, Pop music is particularly regular in its structure, which simplifies the problem compared to the segmentation of any musical piece from any musical genre.

Contributions in this thesis mainly cover three axes: barwise processing of music, a segmentation algorithm called CBM, and the use of compression methods to analyze musical structures. These axes are detailed hereafter.

1. Barwise processing of music. In this thesis, we assume that the barscale is the most relevant scale to study structural segmentation, in particular for Pop music. Indeed, we assume that structural boundaries can be placed on downbeats, *i.e.* between two bars. In addition, we assume that musical motifs are developed at the barscale,

and our methods are designed so as to catch these barwise motifs, thus defining segments according to these motifs and their local redundancies.

Barwise processing is performed on the input data: by computing the spectrogram of a song (its feature description), and by simultaneously estimating downbeats in this song (by using the *madmom* toolbox [Böc+16], which is external from our work), our work consists of splitting the single spectrogram as a collection of barwise spectrograms. This process is detailed in Figure 1.1, resulting into two representations for a song: the **Barwise TF** matrix, introduced in Chapter 2, which regroups both frequency and inner-bar time dimensions in only one dimension, and the **TFB tensor**, a third-order tensor (tensors are introduced in Chapter 4).

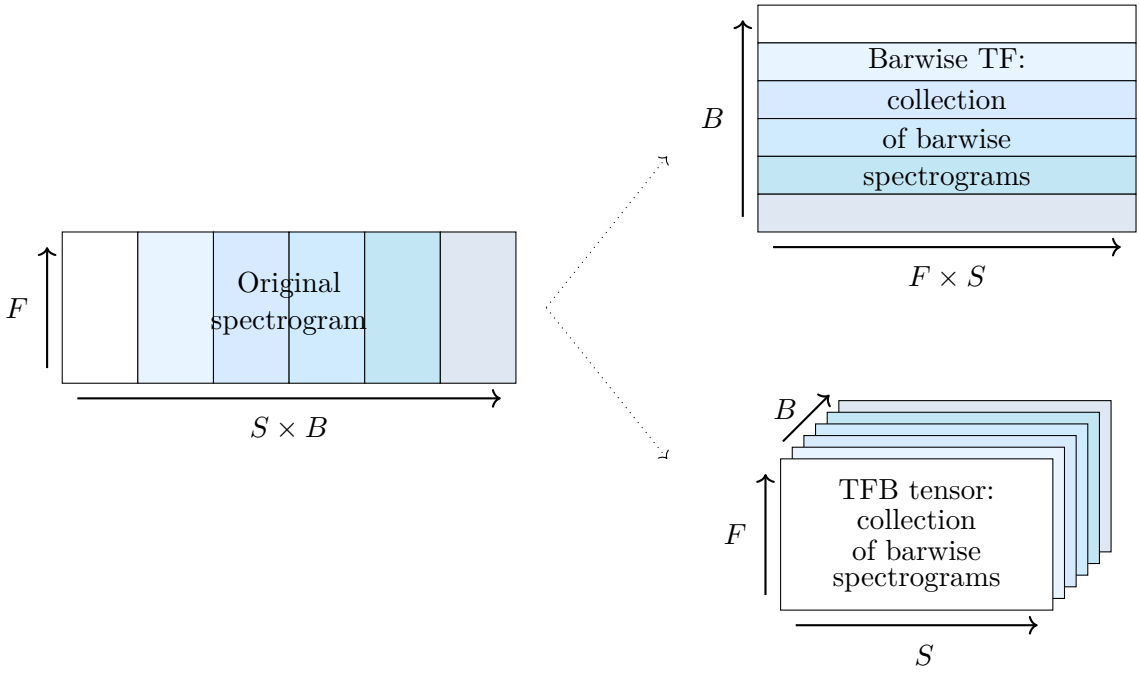


Figure 1.1 – Barwise processing of music, resulting in a matrix (the Barwise TF matrix) and in a tensor (the TFB tensor), depending on the technique which is to be used. Dimensions F, S, B are represented so as to ease the understanding of the process, but are not at the same scale in the different representations.

2. A new segmentation algorithm, called "CBM" for *Convolutive Block Matching*. This algorithm is a dynamic programming algorithm, *i.e.* an algorithm which divides an initial task (here, structural segmentation) in a set of subtasks (here, a score associated with each potential segment). Solving the subtasks recursively (in an optimization scheme) results in a solution for the initial task. Hence, in the CBM

algorithm, the structural segmentation problem boils down to the definition of a score for each segment, and the dynamic programming computes an optimal global solution for this score at the song scale (maximization), as introduced in [Jen06; SBV16]. The main contribution related to the CBM algorithm is the definition of the score function applying on each segment. In our work, the score function focuses on the homogeneity/novelty and on the regularity criteria, as detailed in Chapter 3.

3. The use of compression schemes on barwise representation of the song. In this thesis, we apply several compression schemes on the barwise representations of music. In particular, we focus on the Nonnegative Tucker Decomposition (NTD), which is a tensor factorization scheme sketched in Figure 1.2 and detailed in Chapter 4. The thesis also considers alternative matrix compression schemes, namely Nonnegative Matrix Factorization (NMF), Principal Component Analysis (PCA) and AutoEncoders (AE), detailed in Chapter 5.

   It is worth noting that AutoEncoders studied in this thesis are not designed so as to learn general embeddings on a large dataset, but are rather used as compression schemes at the song scale, *i.e.* independently compressing each song, which, to the best of our knowledge, has not previously been studied for music.

   Finally, this thesis investigates a mix between both NTD and AutoEncoders paradigms, coined "AE-NTD", and detailed in Chapter 6. While some works previously mixed AutoEncoders with standard factorization methods such as NMF [SV17] and the CANDECOMP/PARAFAC decomposition [CCS19], we believe that this chapter represents an original work mixing NTD and AutoEncoders, inspired from these previous developments.

The outline of this thesis is schematized in Figure 1.3. In addition to the structural segmentation task, NTD and AE-NTD are studied on a task of pattern uncovering, consisting of estimating the audio quality of the barwise factorization outputs when transformed into audio signals. This task aims at evaluating the interpretability of both factorization schemes.

What are the experimental hypotheses of this thesis? The various methods are studied in the context of structural segmentation, with quantitative results on the RWC Pop [Got+02] and SALAMI [Smi+11] datasets. The goals of these experiments are 1) to evaluate the CBM algorithm, 2) to study to what extent the use of compression schemes

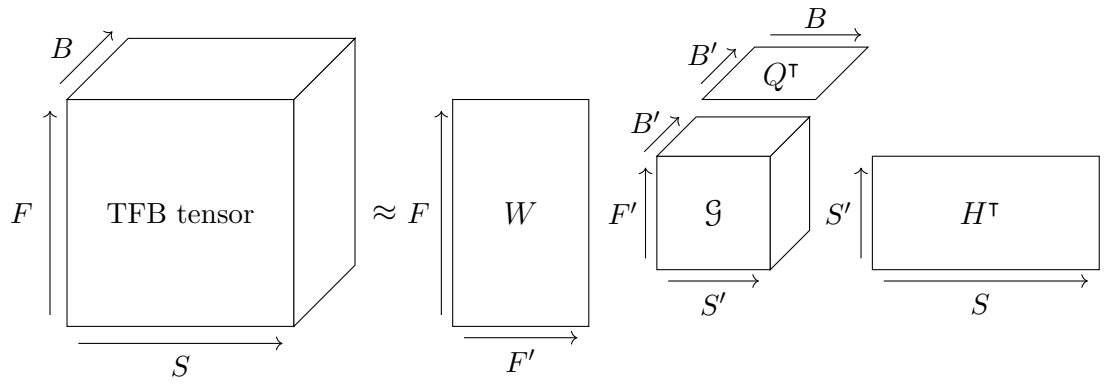


Figure 1.2 – Nonnegative Tucker Decomposition (NTD) of the TFB tensor, *i.e.* the tensor composed of barwise spectrograms. NTD results in three factor matrices W, H, Q, and a "core" tensor $\mathcal{G}$.

is relevant to the task of structural segmentation, and 3) how the choice of the similarity function used to compare the different bars influences segmentation results.

Mathematical notations Throughout this thesis, we use mathematical objects, and define hereafter the associated mathematical notations.

— Vectors are denoted as lowercase letters, *e.g.* a.

— Matrices are denoted as capital letters, *e.g.* A. Rows of a matrix, which are hence vectors, are denoted with their index as subscript, *e.g.* A_i, and equivalently for the columns $A_{:i}$, where the colon is a notation for "all the values on this dimension".

— Third-order tensors are denoted as Euler letters, *e.g.* $\mathcal{A}$. Colons are also used to represent particular dimensions of a tensor, *e.g.* $\mathcal{A}_{:i:}$ represents every element where the index of the second dimension is equal to i.

— The elementwise product between vectors/matrices/tensors, *e.g.* a and b, is denoted as $a.b$, and the elementwise division as $\frac{a}{b}$.

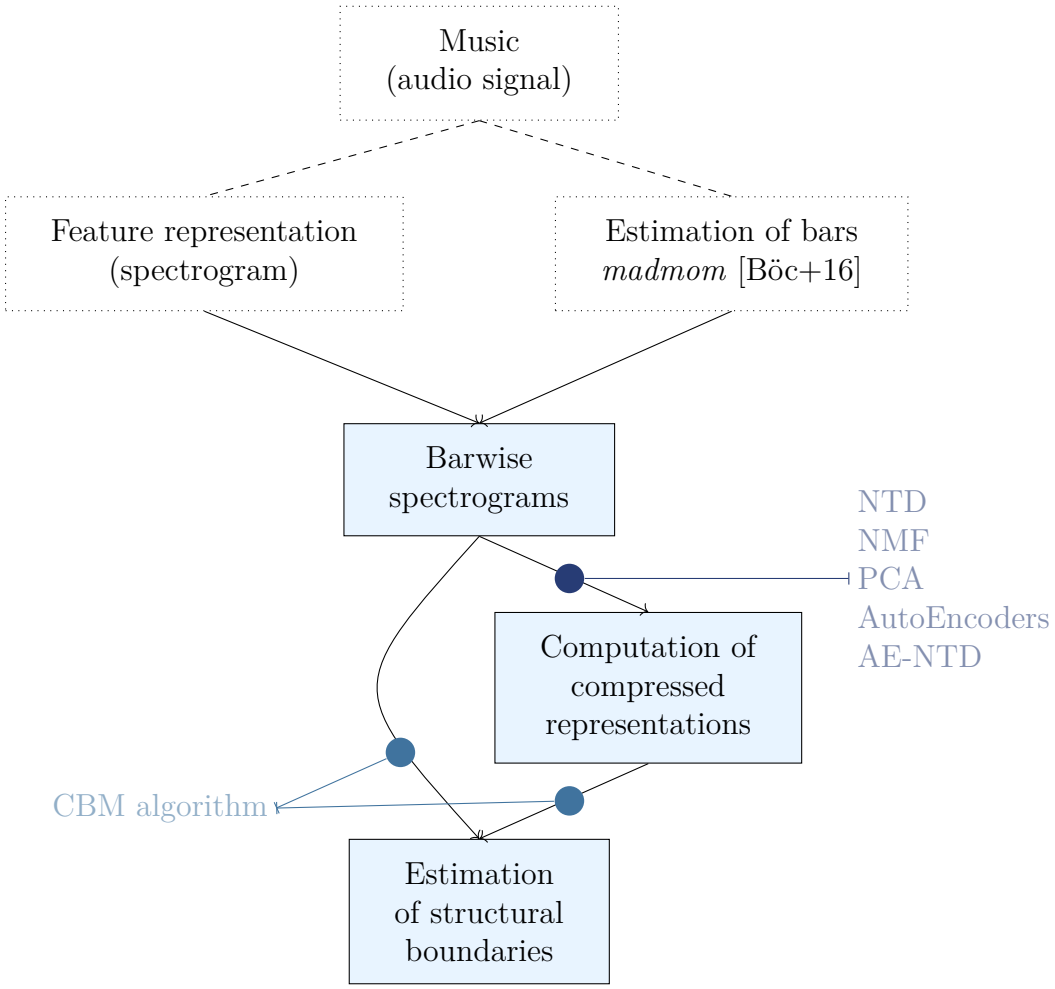


Figure 1.3 – This thesis covers the computation of barwise spectrograms, the CBM algorithm for estimating structural boundaries, and the impact of using compressed representations on the estimation quality. *The stages of feature representation and bar estimation, while important to the process, use earlier work from the MIR community, and do not constitute novel contributions in this thesis.*

Three open-source toolboxes were developed during this PhD, each dedicated to a part of the work: the CBM algorithm is included in the *as_seg* toolbox [MCB22a], the NTD and NMF resolution algorithms are included in the *nn_fac* toolbox [MC20], and the entire work related to the compressed representations is pooled in the *BarMusComp* toolbox [MCB22b].

Part I

Music Structural Segmentation

Chapter 2

Music Processing and Music Structure Analysis

Synopsis

This chapter presents the essential tools to study music computationally, the standard algorithms for studying structure in music, and important concepts used throughout this thesis, such as bar-wise analysis of music.

2.1 Introduction

In a first part, this chapter aims at presenting basic and important music theory concepts, though in a simplified manner. Indeed, while this thesis studies the musical object, presented music theory concepts are intended to properly introduce technical aspects of the work, and not to contribute to the field of musicology. In addition, this chapter also introduces important tools for the computational processing of music, particularly time-frequency representations of music, regrouped under the concept of "spectrogram".

In a second part, this chapter presents the task of Music Structure Analysis, and in particular the subtask of boundary retrieval ("structural segmentation"), along with a review of the literature in this domain. Finally, this chapter introduces in a third part the concept of barwise processing of music, which is at the heart of this thesis.

2.2 Music Processing (generalities)

2.2.1 Audio and Symbolic Representations of Music

Music is perceived by humans in the form of audio signals, via acoustic waves, and is generally conveyed in this format, whether on vinyls, CDs, streaming platforms, or many other media supports. An example of audio signal is presented in Figure 2.1.

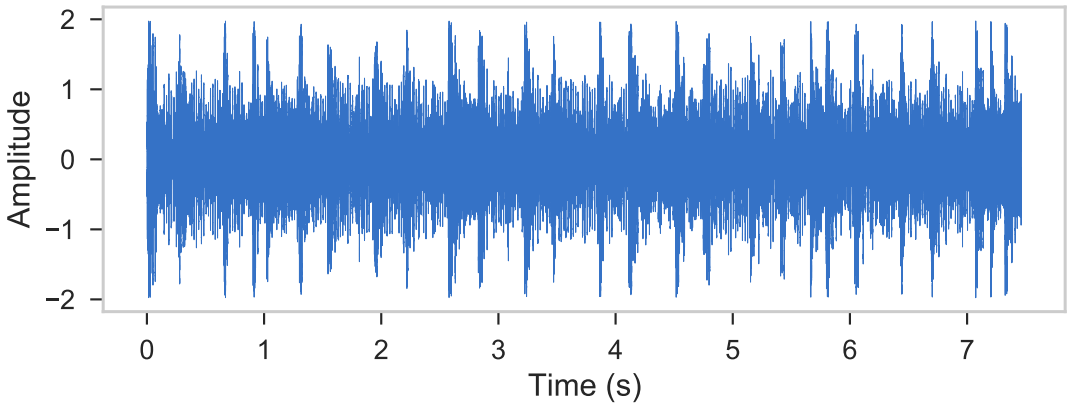


Figure 2.1 – An example of audio signal: passage from the song *Rosetta Stoned* of the band TOOL.

Even though music may originate from human actions or natural events not necessarily designed for musical purposes, as for instance in "musique concrète" or in "sampling", for a

large part music can be considered as intentionally designed by musicians, with or without musical instruments [1]. Hence, to help the transmission of information between musicians, humans developed abstract music notations, describing the audio musical content.

Citing Neely, "music notation is a recipe" [Nee22], particularly useful for learning an instrument, performing a song or composing a music piece. The Western musical score [2] is an example of human-intended music notation, with an example presented in Figure 2.2. More generally, we can refer to music notation as the "symbolic" representation of music. Originally designed for human, the symbolic representation of music also encompasses computational formats, such as the Musical Instrument Digital Interface (MIDI) format [3].

Symbolic representations of music rely, in their vast majority, on the musical "note" as the basic element of music. Each musical note represents a musical event, and the composition of notes form the final score. In the audio domain, musical notes are mainly defined according to 4 parameters [Ber09], namely the **pitch**, the **duration**, the **loudness** and the **timbre**.

The pitch of a note allows to place it on a frequency-related scale, from "low" to "high" note. Depending on the musical instrument, the pitch can be more or less well defined, but, in a vast majority of cases, it is related to the fundamental frequency of the sound, denoted as f_0. The fundamental frequency additionally defines "harmonic partials": the multiples kf_0 of the fundamental frequency.

The duration of a note is its temporal location, generally defined as an onset and a length. The loudness of a note corresponds to its "volume", and is related to the amplitude of the signal. Finally, the timbre models pretty much everything not covered by the other three dimensions, and is hard to define precisely. Figuratively, the timbre primarily is what differentiates two notes of same pitch, duration and loudness, but played on two different instruments. It is related to the spectral and temporal envelopes of the signal, and influences the amplitude ratios between partials and their evolution.

In a musical score, such as the one presented in Figure 2.2, only the pitch, duration

1. The human voice can be considered as a musical instrument.

2. Music in general, and musical notations subsequently, are strongly rooted in a cultural context. This thesis focuses on the standards of the Western culture, defined in Europe around the 16^{th} century, as it constitutes the cultural background of the authors, and as these standards are generally used as "music theory standards" nowadays. Nonetheless, note that other cultural practices and standards exist, and should be studied for a more exhaustive view on "music".

3. Some authors, for instance Müller [Mül15], make the distinction between the visual representations of a score (*i.e.* the sheet music), and the symbolic format, which should explicitly model each musical event, *i.e.* broadly, human- and machine-intended representations. We do not make this distinction here, and consider both formats as symbolic, in opposition to audio representations.

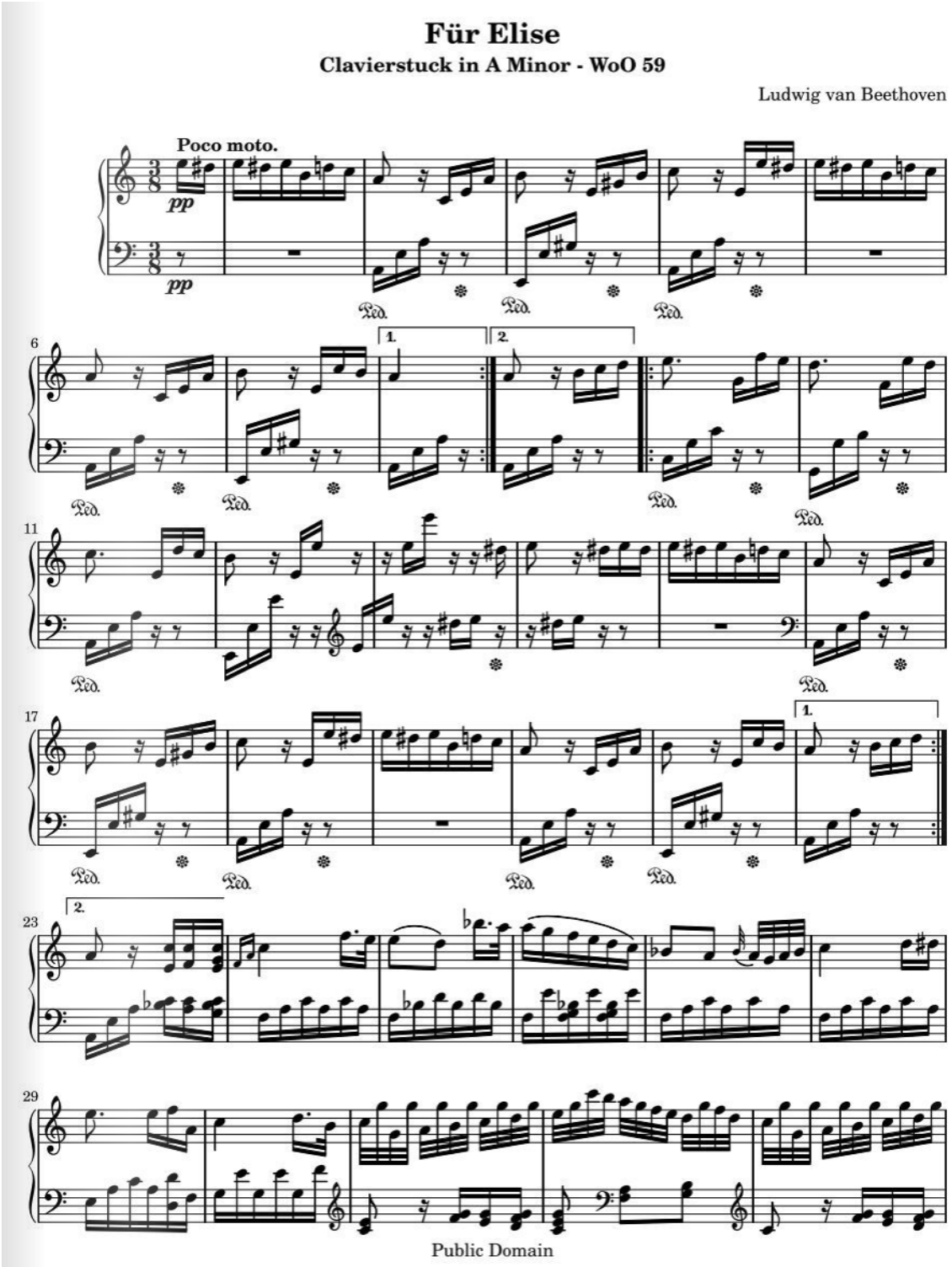


Figure 2.2 – Musical score example

and loudness of a note are represented. In particular, the vertical position of the note represents its pitch on the "chromatic scale", defined hereafter, relatively to the "clef" which indicates the position of the pitch of reference; the shape of the note represents the duration, as detailed in Section 2.2.2. The loudness is indicated as a symbol under the note, such as *pp* under the first note in Figure 2.2. Loosely speaking, several notes, when played simultaneously, define a "chord".

In this thesis, we focus on the audio representation of music. In particular, we study music according to two dimensions: the frequency dimension (relating to the pitch and timbre of notes) and the time dimension, detailed in Section 2.2.2. While the note is defined here for the symbolic representation of music, it has strong connections with the audio representation and interpretation of music.

Chromatic scale

The Western pitch scale is generally reduced to the chromatic scale in equal temperament, which is composed of 12 semi-tones and their octaves. In details, this scale is constructed starting from the 7 notes of the C-major scale: A, B, C, D, E, F, G. Notes on this scale are either spaced by a tone (*e.g.* A and B) or a semi-tone (*e.g.* B and C), leading to a scale composed of 5 whole tones and 2 semi-tones. The 5 remaining semi-tones are obtained by adding a flat (♭) or a sharp (♯) symbol to a note (*e.g.* A♯), which respectively decrease or increase the pitch by a semi-tone. The equal temperament is enharmonic, meaning that the chromatic scale is exactly discretized in semi-tones, *i.e.* increasing a note by a semi-tone results exactly in the upper note (*e.g.* A♯ = B♭ and B♯ = C). Finally, the octave of a note represents the same note, increased by 12 semi-tones. An octave is represented by adding a number to the note (*e.g.* C4).

2.2.2 Time in Music

The most basic unit of time in music is referred to as "pulse". According to Cooper and Meyer, a pulse is "one of a series of regularly recurring [...] stimuli" [CM63]. In that sense, pulses are regular events occurring in a larger information context. In music, pulses

can be seen as the smallest regular division of time, at which people tend to clap their hands or snap their fingers.

Starting from the pulses, Cooper and Meyer then defines the meter as "a measurement of the number of pulses between more or less regularly recurring accents". The notion of accents means that some pulses are accented, compared to others, and that these accents organize and differentiate the different pulses. The meter hence determines which pulses are important in the context of this music, even if these pulses are not sounded.

This leads Cooper and Meyer to define one of the most important notion in music, which is the **beat**, as "pulses [...] counted within a metric context". Beat is an important notion in Western music, as a reference point for musicians and listeners when performing and/or listening to music. Beats may be exactly all pulses, or only some regularly spaced pulses. The frequency of beats generally serves as a reference for the tempo, as in the "bpm" metric (beats per minute).

The "metric context" is the information about the organization of beats and accented pulses at a larger time scale, particularly the **bar**. The bar (or measure) is typically a larger counterpart of beat in Western music, defining the segments where patterns tend to develop and repeat. The bar segments the song, and beats within a bar are ordered (with beat "1" as the start of a bar). We define the **downbeat** as the start of the bar, *i.e.* the beat "1" of a bar.

In Western music, time is generally expressed in beats and bars, and notes are expressed relatively to them, via note values and time signatures. Time signatures are composed of two numbers, one being placed on top of the other, such as $\frac{4}{4}$. The top number indicates the number of beats in a bar, and the bottom number indicates the note duration of each beat. With these two numbers, the reader knows the duration of each beat and the number of beats in each bar.

Finally, the duration of each note is expressed by a note value. The basis unit of time for a note is the "whole note" (or "semibreve"), represented by a 1 in the time signature. This whole note is then divided to obtain "half notes" (2 in the time signature, half the duration of a whole note), "quarter notes" (4 in the time signature, a quarter of the duration of the whole note), etc. The different note values are presented in Figure 2.3.

Hence, given the time signature, a trained musician can understand the rhythmic organization of the different notes composing a musical score, and can play the song [4].

4. A musician would also need an additional information about tempo to accurately perform a song, for instance in bpm, indicating the number of beats per minute, to represent the duration of a beat in absolute time.

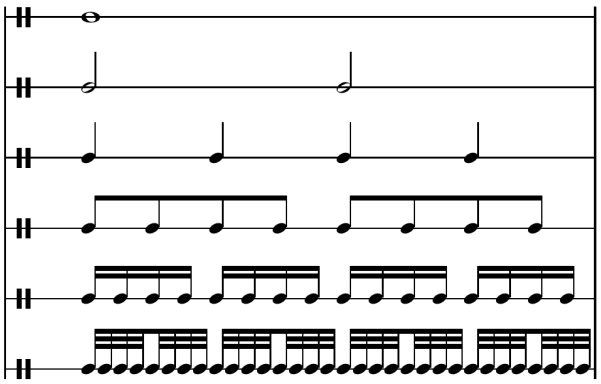


Figure 2.3 – Note values, from top to bottom: a whole note, half notes, quarter notes, eighth notes, sixteenth notes and thirty-second notes. Each line represents the same amount of time. (From Wikimedia, Public Domain.)

In Western music, time is organized around the two concepts of beats and bars. This is particularly visible in musical scores, as the previous example presented in Figure 2.2: in this example (and in musical scores in general), bars are represented by vertical lines (or vertical "bars"), and notes are expressed by their note values, relatively to the beats, the latter being defined relatively to the time signature. In this particular example, the time signature is $\frac{3}{8}$, meaning that each bar contains three beats, and each beat lasts one eighth note. In this example, almost every bar contains six sixteenth notes, meaning that each note lasts half of a beat.

> Summing up, in a musical score, the time is divided in bars, themselves divided in beats. Given the clef (reference pitch) and the time signature (reference duration), the symbol of a note expresses its pitch and its note duration, respectively in its vertical position and its shape.

The remaining of this thesis focuses on audio representations and not symbolic ones, such as the score, but the musical score is at the heart of musical understanding and composition. In particular, the score of a music piece can be seen as a compressed representation of its audio content. Thus, music analysis in this thesis can be understood in this viewpoint: interpreting audio signals as individual notes (or chords), located at precise locations in a bar, and using compression methods in order to exhibit the motifs arising from arrangements of these notes. This will be in particular the case for "musical patterns", latter introduced in Section 4.4.

2.2.3 Digital Signal Processing

An audio signal is an acoustic wave, originating from one or several sources, generally musical instruments, and recorded by microphones. In that sense, a music audio signal can be represented as a continuous function x such that:

$$\begin{array}{rccc} x & : & \mathbb{R} & \rightarrow & \mathbb{R} \\ & & t & \mapsto & x(t), \end{array} \tag{2.1}$$

t representing the time instances and $x(t)$ the amplitudes. In practice, computers are not suited to handle non-stationary continuous functions, like audio recordings. Thus, digital signal processing requires additional stages of **sampling** and **quantization** [Mül15], respectively aiming at discretizing the analog signal in time and amplitude.

In the time domain, sampling consists of evaluating the signal on a finite subset of time instances, as represented in the upper part of Figure 2.4. In practice, denoting as T the sampling period, *i.e.* the gap in time between two consecutive samples, sampling is obtained by evaluating x on the subset $\{nT/n \in \mathbb{N}\}$, finite when the signal is itself of finite duration. Hence, we define the sampling operator S in Equation 2.2 [5]:

$$\begin{array}{rccc} S & : & \mathbb{R} & \rightarrow & T\mathbb{N} \\ & & t & \mapsto & T\lfloor \frac{t}{T} \rfloor. \end{array} \tag{2.2}$$

Sampling can be obtained by the composition $x \circ S$.

In general, sampling is expressed according to the **sampling frequency** $f_s = \frac{1}{T}$, *i.e.* the number of frames sampled for a given unit of time. When expressed in Hz, the sampling rate represents the number of samples per second. A standard value for the sampling rate in music is $f_s = 44.100$Hz, typically used for compact discs. This standard value satisfies the Shannon sampling theorem [Sha49].

5. We define the notation $k\mathbb{N}$ as the set of integers multiplied by the constant k.

Shannon sampling theorem

The Shannon sampling theorem states that the sampling rate f_s must be at least twice as much as the largest frequency of the original continuous signal, *i.e.* denoting as F the highest frequency in the signal, $f_s \geq 2F$.

As the human hearing range (*i.e.* the range of audible frequencies for a general human ear) ends at 20kHz, higher frequencies are of no use in this context, and, as a consequence, validates the Shannon sampling theorem for $f_s = 44.100$Hz.

In the amplitude domain, quantization consists of representing the amplitude values on a discrete grid, as represented in the lower part of Figure 2.4. The grid of discrete values is defined by a quantization step A, analogous of the sampling period in the amplitude domain. Hence, starting from a continuous signal $x(t)$, we present in Equation 2.3 an example of quantization operator Q defined in [Mül15]:

$$\begin{array}{rccc} Q \circ x & : & \mathbb{R} & \to & A\mathbb{Z} \\ & & t & \mapsto & \text{sign}(x(t))A\lfloor \frac{|x(t)|}{A} + \frac{1}{2} \rfloor. \end{array} \tag{2.3}$$

Practically, the operator Q takes the value in the grid closest to the original value of the signal. In music processing, for instance for compact discs, it is standard to represent the amplitude values with 16bits, leading to 65.536 discrete values to quantize amplitudes [6].

Hence, a digital audio signal can be seen as a function of discrete time and amplitude values, *i.e.* a function from $T\mathbb{N}$ to $A\mathbb{Z}$.

2.2.4 Feature Representations

Nowadays, some recent neural networks architectures directly treat the digital audio signal as input, called "end-to-end" networks [DS14], see the recent overview on the use of neural networks in MIR given by Peeters and Richard [PR21, Chap. 3]. Still, the vast majority of work in MIR (and notably the non-deep learning approaches) is unsuited to treat the raw audio signal, and requires a feature transformation, called time-frequency

6. The exact quantization step A also depends on the range of admissible amplitudes (in particular maximal and minimal values) which, to the best of our knowledge, is not subject to a consensus.

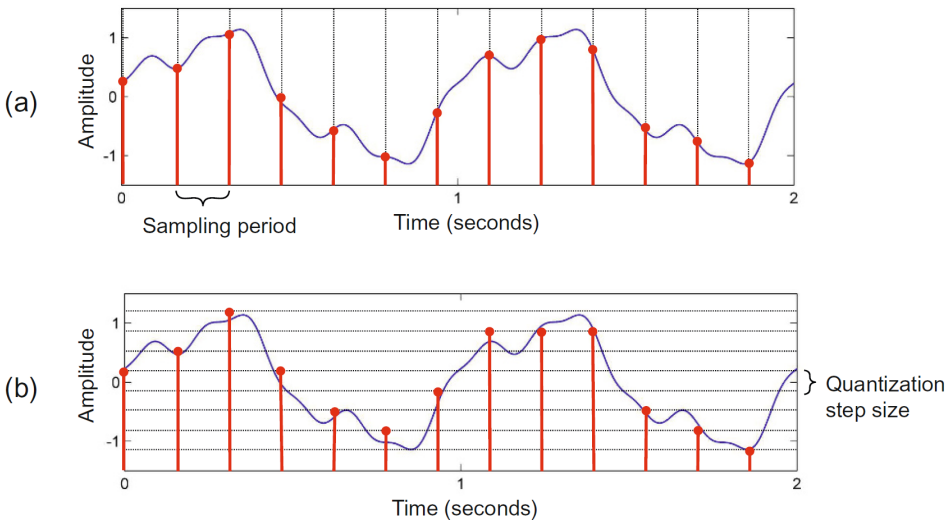


Figure 2.4 – Sampling operation ((a), upper) and quantizing operation ((b), lower), extracted from [Mül15].

analysis, resulting in **time-frequency representations**, abusively referred to as **spectrograms**. We present in this section some of the most important features, of particular interest in this thesis. Details on the computation and parametrization of these features in the context of this thesis are presented in Appendix A.1.

Short-Time Fourier Transform A renowned time-frequency representation of the signal is the Short-Time Fourier Transform (STFT). The rationale of the STFT is to study the frequencies present in the signal, as both pitch and timbre relate to frequential aspects in the waveform. Frequencies are estimated by the discrete Fourier transform applied on frames (*i.e.* blocks of samples) of the digital signal.

The tricky part in the STFT is to compute the Fourier transform on frames long enough to be accurate in the estimation of frequencies (the longer the signal, the more accurate the estimation), but short enough to be accurate in time, given that the signal is non-stationary. Simultaneous perfect time and frequency estimation are impossible, but inconsistencies are generally reasonable for music analysis. Still, some recent works propose to perform Fourier analysis with frames of different sizes at the same time, in a "multiscale" setting [ZEH16] (though it is not exactly the STFT). We restrict this thesis to the study of fixed length frames.

Formally, we denote N the size of each frame, and each frame is windowed by a non-

negative function $w()$, a particular example being the rectangular function [7]. A standard value for N in MIR is 2048, *i.e.* 2048 samples per frame (for signals sampled at 44.100Hz).

The time difference between two consecutive frames in the computation of the STFT is called the "hop length", denoted as H here. In general, $H < N$ in order to allow some overlap between two consecutive STFT frames and limiting side effects.

Finally, the STFT of a signal x is obtained by applying a Dicrete Fourier Transform on each frame, as presented in Equation 2.4. Parameters f and t are respectively the indexes for the frequential and time bandwidths of the resulting STFT, which are evenly spaced. A reader unfamiliar with the STFT, and curious about details, can refer to [Mül15, Chap. 2].

$$STFT(x)(f,t) = \sum_{n=0}^{N-1} x(tH+n)w(n)e^{-\frac{2i\pi fn}{N}} \tag{2.4}$$

STFT results in a complex-valued time-frequency representation, called **STFT spectrogram**, but music analysis is generally performed on real-valued representations, either the modulus $|.|$ or the squared modulus $|.|^2$ of these coefficients, respectively defining magnitude and power spectrograms. Indeed, the modulus (or squared modulus) of the complex-valued coefficients represents the relative importance of each frequency bin at each time bin.

In most of the work using the STFT, the phase information, representing the argument of each complex-valued coefficient, is not considered, though recent work integrates it in the analysis [PR21].

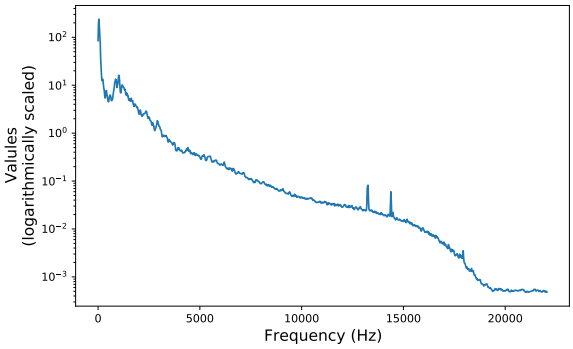
In practice, low-frequency STFT coefficients are generally observed to be larger in magnitude than high-frequency coefficients. This property is exhibited in Figure 2.5a, which presents the average values of the coefficients in a magnitude spectrogram according to the frequency bins. In this example, the decrease in magnitude values is empirically close to logarithmic.

In addition, in audio signals, components of low intensity can perceptually be as important as components of high intensity, due to the perception of loudness in the human ear. In particular, the loudness of a sound is known to depend on the frequency of the sound, especially for low-frequency components. This leads researchers to study the "equal-loudness contours", consisting of evaluating the differences in loudness perception according to the frequency of the sound, see [ST04] for a comparison between models of

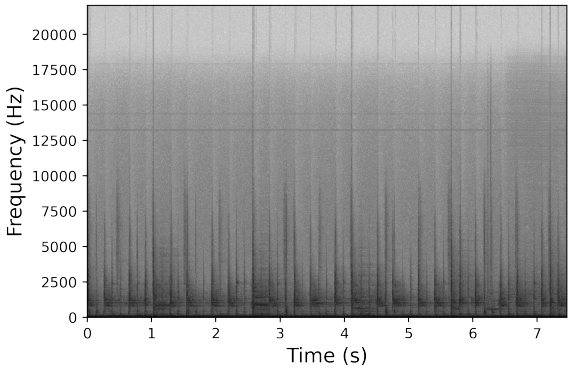
7. 1 for samples at indexes lower than N, 0 otherwise.

equal-loudness contours. Thus, the intensity of a sound and, consequently, dynamics in amplitude, are not proportional to its perceptual evaluation.

In MIR, both of these effects may obscure relevant information and complicate the extraction of information from the raw magnitude spectrogram. For that reason, magnitude spectrograms can be studied by applying the logarithm function to the original coefficients, hence resulting in log-magnitude spectrogram, an example being presented in Figure 2.5b.



(a) Average value of the magnitude coefficients according to each frequency bin (in the magnitude spectrogram).



(b) Logarithm values of the magnitude spectrogram (log-magnitude spectrogram).

Figure 2.5 – Example of STFT magnitude spectrogram, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1). The left figure presents the average magnitude values according to the frequency bins. The right figure presents the log-magnitude spectrogram, *i.e.* the logarithm of the magnitude values of the STFT.

Mel-scale Representations Frequencies on the STFT spectrogram are sampled on a linear scale. In practice though, in an music audio signal, the fundamental frequencies of the different notes are not linearly but exponentially spaced. This is due to the harmonicity of musical notes: starting from a note with a fundamental frequency f_0, its frequential spectrum is principally composed of harmonic partials of multiple frequencies, *i.e.* kf_0. This property is in particular true for the octave of this note, with a fundamental frequency $2f_0$, and its harmonic counterparts $2kf_0$. The second octave for this note (*i.e.* the octave of the octave) have a fundamental frequency $2 \times 2f_0 = 2^2 f_0$, and similarly for the subsequent n octaves with fundamental frequencies $2^n f_0$, and harmonic partials of frequencies $2^n k f_0$.

In that sense, some authors adapted the frequency bins to the spread of the fundamental frequencies of notes, resulting in scales such as Mel scales [SVN37; War70] or

the Equivalent Rectangular Bandwidth (ERB) scale [GM90]. We restrict this study to Mel scales, which not only depends on the aforementioned exponential spread of octaves, but also depends on perceptual evaluation of pitches [SVN37]. The perceptual evaluation phase is intended to space Mel coefficients according to what listeners consider to be an even scale of notes, which may not be exactly exponential in frequencies.

In particular, while several Mel scales coexist in literature [UCN99], we use the particular Equation 2.5, defined by O'Shaughnessy [OSh87], which converts a frequency f in Hertz into a Mel coefficient m as:

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right). \tag{2.5}$$

This conversion is implemented is the *librosa* standard toolbox [McF+21]. Following the Mel conversion, **Mel spectrograms** are computed by keeping only a few anchor points, linearly spaced on the Mel scale, and aggregating the energy of the STFT coefficients around these anchor points.

In practice, this is made by constructing a Mel filter bank (an illustrative example being shown in Figure 2.6) and applying the filter bank to the STFT coefficients. Each filter (here, of triangular shape) considers a few STFT frequency bins only (the number of which depends on the frequency: the higher is the frequency, the more STFT bins are considered), weights each STFT coefficient (depending on the value of the filter at this frequency bin), and sums all resulting coefficients.

The Mel spectrogram, in addition from being more closely related to the acoustic properties of music, also has the advantage of being more efficient computationally, as it reduces the number of features.

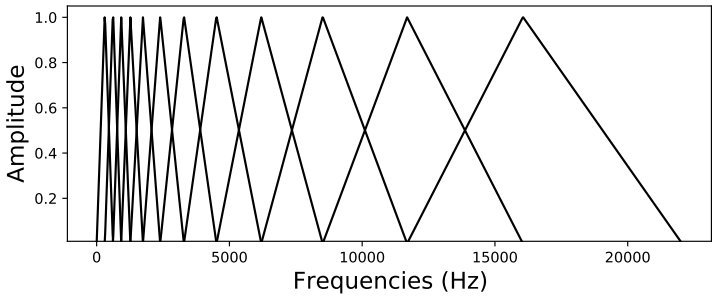


Figure 2.6 – Illustrative example of a Mel filter bank, with only 12 Mel filters, presented on the frequency scale. STFT coefficients are aggregated along the filters.

Mel coefficients being computed from the STFT coefficients, they are subject to the phenomenon presented in Figure 2.5, *i.e.* nonlinear relations between the values of the Mel coefficients and their human perception. In that sense, some authors have applied the logarithm function to the Mel coefficients, resulting in the **Log Mel spectrogram**. In the Log Mel spectrogram, the energy discrepancies leading to small and large Mel components are constricted.

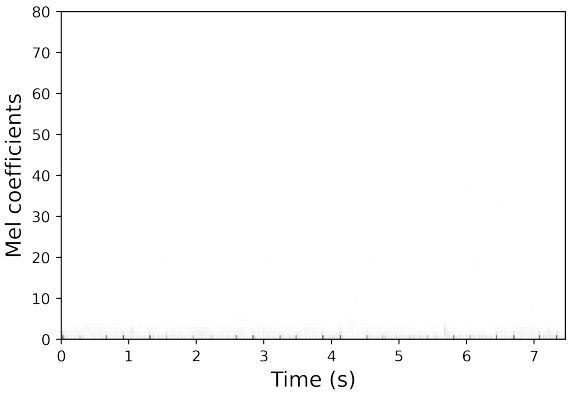
In addition to the previous Log Mel spectrogram, we introduce the **Nonnegative Log Mel spectrogram**, abbreviated **NNLMS** or **NNLM spectrogram**. The NNLMS is computed as $10 \log_{10}(Mel+1)$ where Mel represent the coefficients of the Mel spectrogram, which are nonnegative, and log the elementwise logarithm.

The NNLMS has the advantage of penalizing less the very low Mel coefficients compared to the Log Mel spectrogram, because the logarithm is approximately linear for values close to 1. In that sense, low Mel coefficients remain low in the NNLMS while higher values are damped. In addition, this feature is nonnegative, which will be important for some of the work presented, in particular NTD in Chapter 4.

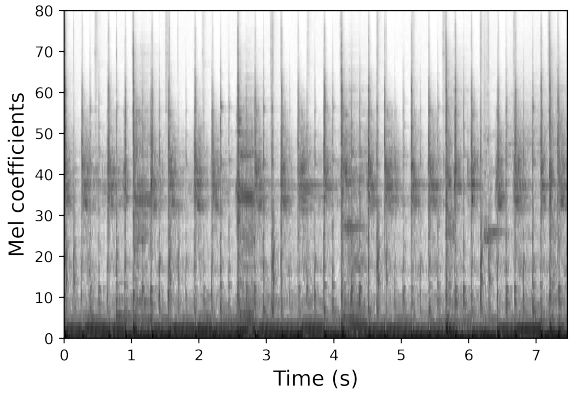
The three spectrograms (Mel, Log Mel and NNLM spectrograms) are presented in Figure 2.7.

Chromagram A **Chromagram** represents the frequency according to the 12 semi-tones of the Western chromatic scale. In this feature, each row represents the weight of a semi-tone (and its octave counterparts) at a particular instant. The rationale of chromagrams is to represent the energy associated with each semi-tone. Originally, chromagrams were computed from a Fourier transform of the signal, by pooling frequency coefficients along pitch-designed bandwidths [Got03], but recent work makes use of deep learning schemes to precisely compute chromagrams [KW16], see [PKS22] for a comparison between recent models.

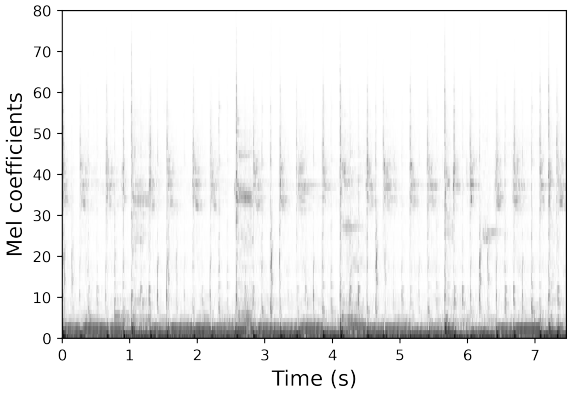
Of particular importance in this thesis is the "Chroma Energy Normalized Spectrogram" (CENS) [MKC05]. After the computation of chroma vectors, the CENS applies a l_1 normalization on each chroma vector, and samples their amplitudes based on "log-like" amplitude thresholds. The latter log-like thresholding is motivated in order to account for the perceptual evaluation of energy discrepancies, as discussed for the Log-magnitude STFT spectrogram. In addition, a smoothing window is applied on the time axis. We abusively use the term chromagrams to denote the particular CENS computed here. An example of chromagram is presented in Figure 2.8.



(a) Mel spectrogram (values are concentrated on the lowest frequencies).



(b) Log Mel spectrogram.



(c) NNLM spectrogram.

Figure 2.7 – Example of Mel, Log Mel and NNLM spectrograms, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1).

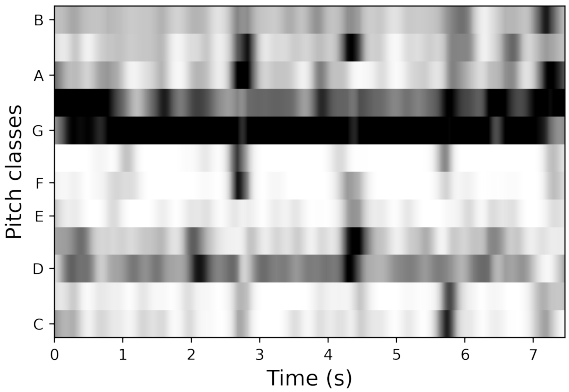


Figure 2.8 – Example of chromagram, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1).

MFCC Mel-Frequency Cepstral Coefficients (**MFCCs**) are timbre-related coefficients, obtained by a discrete cosine transform of a Log Mel spectrogram. The rationale of MFCC is to represent broadly the spectral envelope of a signal at every instant, hence giving information about the evolution of timbre (correlated with the spectral envelope of signals) [PMK10]. An example of MFCC, with 10 coefficients, is presented in Figure 2.9.

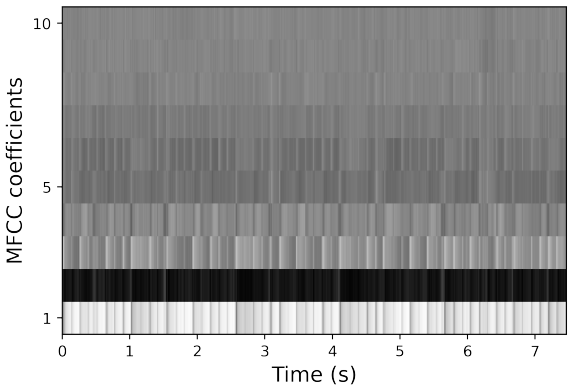


Figure 2.9 – Example of MFCC, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1).

These representations focus on representing the music as a sequence of frequency-related information. On the first hand, STFT, Mel and Log Mel/NNLM spectrograms represent the raw frequential content, estimated by means of Fourier transformations. On the other hand, chromagrams and MFCC focus on a particular aspect of the signal (harmony for chromagrams, timbre for MFCC).

2.3 Music Structure Analysis

Citing Paulus *et al.* [PMK10], "[...] it is the structure, or the relationships between the sound events that create musical meaning". In that sense, researchers in MIR developed the Music Structure Analysis (MSA) task, which focuses on the retrieval of the *structure* in a song.

Music structure is ill-defined, but is generally introduced as a hierarchical system, from the level of notes to the level of the song itself [McF+17; Nie+20]. A tentative definition is that structure is *a simplified representation of the organization of the song.*

In that sense, motifs, which arise from the organization of notes themselves, is a first element of structure. These motifs create phrases and chord progressions, which further define higher-scale elements such as patterns, etc. In general, the highest level of structure defines musical sections, such as "chorus", "verse" and "solo", which is a macroscopic level of the study of music [SBV16]. Some work focus on estimating structure in its hierarchical nature, *e.g.* [ME14a; ME14b; Ber+20; SNB21], but this thesis rather focuses on a "flat" level of segmentation, *i.e.* the macroscopic level, corresponding to musical sections.

MSA is subdivided into two subtasks, not necessarily mutually exclusive: the **boundary retrieval** and the **labelling of segments**.

— The boundary retrieval task consists of estimating the boundaries between different sections, and hence partitions the music in several non-overlapping segments, covering the entire song.

— The labelling task consists of assembling the similar segments altogether by labelling them with similar labels, typically letters such as 'A', 'B', 'C', etc, or plain words such as 'chorus', 'verse' 'solo', etc, *i.e.* musically-designed identifiers corresponding to the role of the different segments in the song. The latter labelling scheme, with musical identifiers, is called "functional" labelling [Bim+12], but works attempting to estimate functional labelling are scarce [WHS22].

A schematic example of musical structure is presented in Figure 2.10. A task is devoted to MSA in the MIREX contest [8].

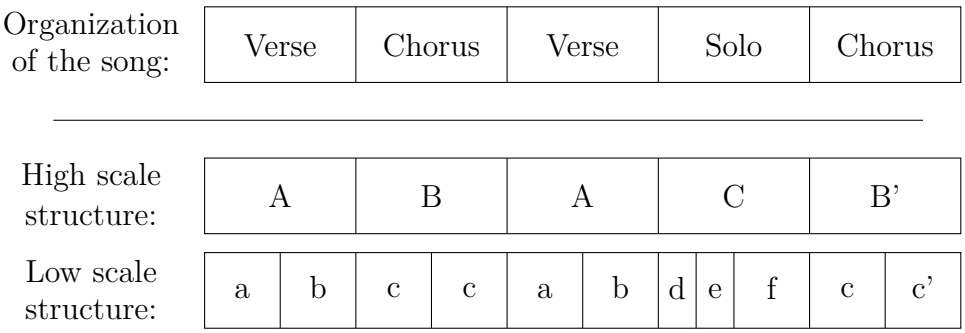


Figure 2.10 – A schematic example of musical structure

Many MSA algorithms make use of matrices representing the similarity and dissimilarity in the music, sometimes referred to as "self-distance matrices" [PMK10], "self-similarity matrices" [Nie+20], "recurrence matrix" or "pair-wise frame similarities" [ME14b]. These

8. *e.g.* www.music-ir.org/mirex/wiki/2016:Structural_Segmentation

representations differ in their details, but share the same conceptual idea of computing a notion of similarity (or, conversely, a notion of distance) between the different frames in the music, and representing it in a square matrix, its size being the number of frames. This representation is also at the heart of our MSA strategy, but we rather use the term of "autosimilarity matrix", as presented in Section 3.2.

In the same spirit, authors introduced "lag similarity matrix" [Got03], which also represents the similarity at the scale of the song, but with time expressed relatively to the current frame rather than as an absolute position in the song. Hence, in this section, we loosely refer to these different representations under the term "autosimilarity matrices".

An idealized autosimilarity matrix, extracted from [PMK10], is presented in Figure 2.11. Similar passages in an autosimilarity matrix lead to 2 typical shapes, exhibited in Figure 2.11: **blocks** and **stripes**. A block is a square (or a rectangle) in the autosimilarity, representing a zone of high inner-similarity, *i.e.* several frames which are highly similar. A stripe is a line, parallel to the main diagonal, representing a repetition of the content, *i.e.* a pattern of several frames repeated in the same order.

The blocks and stripes structures in an autosimilarity matrix depend on the features, which relate to specific attributes in music (*e.g.* timbre for MFCC, harmony for chromagrams, etc), and hence influence the evaluation of similarity and repetition in the song. Blocks and stripes structures also depend on the temporal resolution of the analysis [9] [Pee03], which advocates for musically-motivated temporal divisions, as discussed in Section 2.4.

As a general trend, the algorithms using autosimilarity matrices are designed so as to retrieve segments based on blocks and stripes. Such algorithms are introduced hereafter.

2.3.1 Boundary Retrieval - Structural Segmentation

Formally, given a musical song sampled in time as Ω audio frames, the subtask of boundary retrieval can be defined as finding a set of boundaries Z^e representing the start of all segments, *i.e.* $Z^e = \{\zeta_i^e, i \in [\![1, E]\!]\}$, E representing the number of boundaries estimated in this song. In particular, the first boundary is the start of the song, *i.e.* $\zeta_1^e = 1$, the last boundary is the last audio sample, *i.e.* $\zeta_E^e = \Omega$, and each boundary is located on an audio sample, *i.e.* $\forall i, \zeta_i^e \in [\![1, \Omega]\!]$. The i-th segment S_i is exactly composed of the samples indexed in the interval $[\![\zeta_i^e, \zeta_{i+1}^e[\![$. By definition, E boundaries define $E-1$

9. Note that the author refers to blocks and stripes respectively under then names of "states" and "sequences".

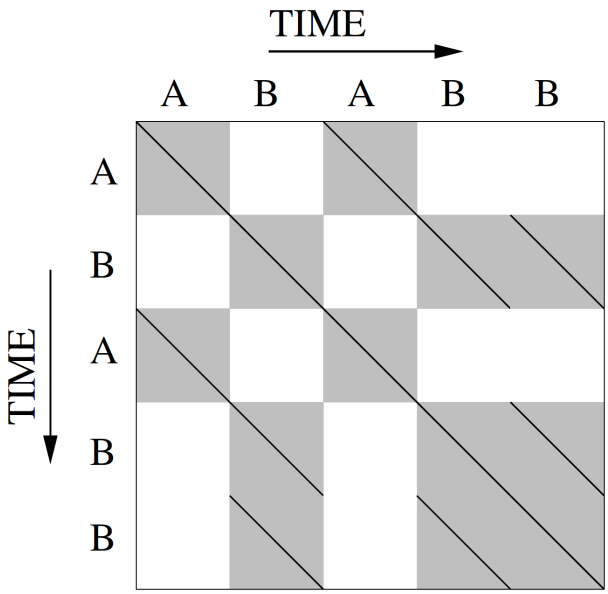


Figure 2.11 – An idealized autosimilarity matrix, extracted from [PMK10].

segments.

Boundary retrieval aims at segmenting the song, and is also referred to as **structural segmentation**. It consists of segmenting the song in non-overlapping parts. The work presented in the subsequent chapters of this thesis focuses on the boundary retrieval task.

Algorithms aimed at solving the task are designed following one or several criteria among four: **homogeneity**, **novelty**, **repetition** and **regularity** [Nie+20]. Homogeneity and novelty relate to one another, and can be studied together.

Homogeneity and Novelty

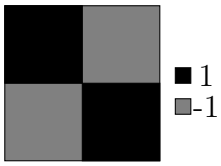
The homogeneity criterion consists of defining a segment as internally stable with respect to some musical attributes, *e.g.* harmonic content or timbre characteristics. The rationale is that similar passages must belong to a same segment, as they share musical properties. The novelty is the complementary criterion, focusing on retrieving boundaries as points of high dissimilarity, for instance a change in instrumentation or a switch to another harmonic line.

As an immediate observation, homogeneity focuses on finding segments themselves, or strongly similar parts in a music, while novelty focuses on the boundaries between segments. In that sense, both criteria are often considered as two viewpoints of the same problem, typically a point of high novelty may be defined as a breaking point of homo-

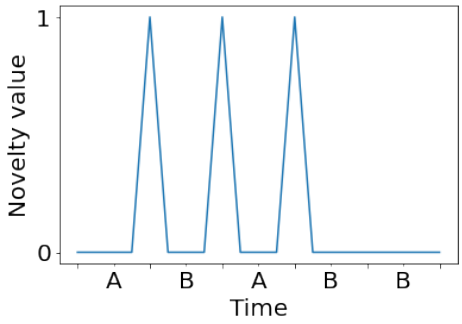
geneity, and homogeneity values are low or high relatively to the novelty values.

The kernel of Foote [Foo00], which may be one of the earliest work on audio MSA, exploits this definition of novelty, and estimates boundaries as points of high dissimilarity between the recent past and the near future. The kernel is a square block matrix, composed of 4 square blocks of same size, either constant equal to 1 or equal to -1. Hence, the 4 blocks divide the kernel in 4 smaller square blocks. The upper left and lower right blocks are constant equal to 1, and the lower left and upper right blocks are constant equal to -1, for instance: $\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$, also represented in Figure 2.12a.

This kernel works ideally when the recent past and the near future are homogenous in their respective neighborhoods, but very dissimilar with each other, for instance on the diagonal point between the first A and B segments in Figure 2.11. In practice, this kernel is convolved with the autosimilarity matrix of the song, centered on the diagonal, which gives a "novelty" value for each temporal sample of the song, finally post-processed into boundaries with a thresholding operation. An illustration of novelty, computed on the idealized autosimilarity matrix of Figure 2.11, is presented in Figure 2.12b.



(a) Example of a binary kernel from Foote [Foo00].



(b) Illustration of novelty computed with Foote's kernel on the idealized autosimilarity matrix of Figure 2.11.

Figure 2.12 – Representation of the Foote's novelty kernel.

The size of the kernel strongly influences the computation of novelty, and hence influences the boundaries. This impact may be tampered by a smoothing operation on the values of the kernel, typically with a gaussian kernel, as introduced in the seminal paper [Foo00], which also smoothes the novelty curve.

While this technique is rather simple, it is still used as a standard segmentation tool in recent work, *e.g.* [McC19; Wan+21] focusing on improving the segmentation scores by en-

hancing the autosimilarity matrix. In particular, both of these works belong to the domain of representation learning [BCV13], consisting of designing machine learning algorithms prone to learn relevant representations instead of focusing on solving a particular task. In that context, using prior knowledge, both works design neural networks architectures and optimization schemes with the objective to obtain enhanced (nonlinear) similarity functions, more prone to highlight the structure in the autosimilarity matrices.

Notably, McCallum [McC19] develops an unsupervised learning scheme where the prior knowledge enforced in the representation is based on the proximity of samples: the closer the frames in the song, the more probable they belong to the same segment. In the same spirit, Wang *et al.* [Wan+21] develop a supervised learning scheme: the neural network must learn representations where segments annotated with the same label are close, and segments annotated differently are far apart. The rationale for both methods is to learn a similarity function which is not only representing the feature-wise correspondence of two music frames, but can also discover frequent patterns in the learning samples, and hence transform the stripes (corresponding to repeated patterns) into blocks, thus favoring homogeneity.

To some extent, this is also the objective of the Spiral Array model [Che02], used for the related task of key segmentation, and implemented for audio-based analysis by Chuan & Chew [CC05]. Indeed, by projecting the chords into a new feature space (here the spiral-array), the musical patterns turn into tonal representations accounting for musical proximity of the chords, hence turning the sequential aspect of music into some form of homogeneity.

Not all techniques focusing on novelty make use of Foote's kernel. Considering homogeneity as blocks, Jensen [Jen06] developed an optimization problem aiming at minimizing the average self-dissimilarity of each segment as a way to score the homogeneity of each segment. This optimization problem is solved by dynamic programming (more precisely, finding the shortest path in a directed acyclic graph), with an additional constraint in order to limit the number of segments. Sargent *et al.* [SBV16] later extended this optimization paradigm to the incorporation of constraints, based on prior knowledge, in particular on the size of segments.

Another, popular, dynamic programming algorithm is the Viterbi algorithm, used in particular to solve Hidden Markov Models (HMM), as detailed in [Rab89]. HMM were used in the context of music segmentation to represent individual segments as states, and the Viterbi algorithm decodes the sequence of states to compute the final segmentation [LC00;

AM01; PLR02].

Clustering methods, *e.g.* k-means or Nonnegative Matrix Factorization (NMF) (NMF is formally introduced in Section 4.2.3), which aims at discovering correlations between data points by regrouping them in large classes ("clusters"), are standard in many machine learning tasks to uncover similarities in data without supervision. One such example in MSA is the use of Convex NMF [NJ13], a variant of NMF were the feature space is contracted in convex combinations of columns of the original data. In the context of structural segmentation, Convex NMF is applied on autosimilarity matrices, and factorization results are thus interpreted as the most similar frames, then processed into sections.

Clustering methods may seem adapted for uncovering sections following the homogeneity criterion. Nonetheless, as pointed out by Peeters *et al.* [PLR02], standard unconstrained clustering methods often fail to catch the temporal continuity in consecutive frames to form sectional clusters, while it is primordial in music. In that sense, Levy & Sandler [LS08] developed a temporally-constrained clustering algorithm, applied on states sequences of a HMM. In that same idea, McFee and Ellis [ME14b] uses a constrained agglomerative clustering to form consistent clusters, where homogeneity is enhanced using supervision.

Repetition

While the homogeneity/novelty criteria consider segments as locally uniform, the repetition criterion considers that segments should be sequences, prone to be repeated in the song. In that sense, individual sequences may be self-dissimilar by nature, *i.e.* composed of very different musical components, but the fact that they are repeated constitutes the segment. This is a common assumption in music when considering musical attributes, like chord progressions in harmony for instance.

This criterion is very popular in Symbolic MSA (retrieving the structure of music seen as a symbolic flow of information, typically chords), see [PMW10] for an overview of melodic segmentation methods, *i.e.* techniques aiming at computing the main motifs and melodic phrases in a music piece. More recently, Giraud & Staworko [GS15a] and Guichaoua [Gui17] developed algorithms searching for musical sequences based on grammars. Guichaoua [Gui17] and Louboutin [LB17] developed additional models based on polytopic graphs, the former model being presented in the Appendix D.2 [10].

10. In addition to the presentation of the model, Appendix D.2 presents the opensource *MusicOnPolytopes* toolbox [MCB21], which has been developed in the context of this thesis, and which regroups both

In the audio domain, and more particularly in the context of autosimilarity-based techniques, the repetition criterion focuses on retrieving stripes in the autosimilarity matrix. In practice, algorithms must pre-process the autosimilarity matrix, to enhance the stripes [PMK10]. Pre-processing may be inspired from image processing, such as average [BW01], median [MND09], erosion/dilation [LWZ04] or additional filtering [Got03; Pee07].

After this pre-processing operation, stripes may be located in the autosimilarity matrix. Dannenberg & Hu [DH03] and Goto [Got03] proposed algorithms based on thresholding to select the most important stripes. While the algorithm by Goto [Got03] initially focused on chorus retrieval, it was later extended for MSA by Ong [Ong+06, Chap. 4].

Shiu *et al.* [SJK06] proposed to detect stripes by applying a convolution kernel, in the same spirit as Foote [Foo00], but specifically designed for stripes. In details, this kernel is a binary kernel, with 1 on several diagonals and 0 everywhere else, which results in a high repetition value when several stripes are present in the autosimilarity of the current segment considered, and a low repetition value when no stripes are present. An example of kernel is presented on Figure 2.13.

In this work, convolution kernels are more particularly used to estimate the period between two consecutive stripes, the algorithm applying different kernels concurrently. Finally, the periods of repetition are used as constraints in an optimization scheme, such that excerpts with different periods of repetitions should not belong to a same segment. The optimization scheme itself is not based on the kernels.

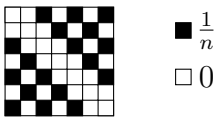


Figure 2.13 – Kernel from Shiu *et al.* [SJK06], designed to find stripes (and so, patterns) repeating every two frames. The parameter n is the number of nonzero elements (18 in this example).

As for homogeneity, some clustering methods have been employed, focusing on retrieving musical sequences. In that sense, Weiss & Bello [WB10] used a Convolutive NMF model to catch repetitive sequences in the music. The Convolutive aspect of NMF allows to incorporate several consecutive frames in the feature space, hence representing patterns of several frames in factors. Convolutive NMF needs a precise number of time frames in

models [Gui17; LB17] in a same framework.

the feature space though, as a parameter, while patterns can be of different sizes. To counteract this effect, the authors implemented sparsity constraints.

In the same spirit, Cheng *et al.* [CSG18] uses an extended version of Convolutive NMF, called "Nonnegative matrix factor 2-D deconvolution" [SM06], which, applied to the autosimilarity matrix, retrieves the stripes structure and, hence, their repetition, finally post-processed into boundaries.

Still focusing on stripes is the work of McFee & Ellis [ME14a], which developed an algorithm based on spectral clustering, aiming at interpreting the stripes as principally connected vertices in a graph. The structure is then obtained by studying the eigenvectors of the Laplacian of this graph, forming cluster classes for segmentation. This technique is amongst the best-performing techniques nowadays, and is improved by recent work of Salamon *et al.* [SNB21], which replace or enhance the acoustic features on which is applied spectral clustering by nonlinear embeddings, learned by ways of a neural network.

Combining Homogeneity and Repetition

While the aforementioned algorithms mainly focus on one criterion from homogeneity/novelty and repetition, the concepts are not mutually exclusive. In that sense, some methods combine the different criteria in a larger framework, a particular example being the constrained clustering algorithm of Levy & Sandler [LS08] that we previously introduced under the homogeneity criterion. Indeed, the time constraints favor the construction of clusters based on temporal proximity, and hence favor clusters based on repetitive sequences rather than clusters based on feature-wise similarity only.

Kauppinen *et al.* [KKV13] present an "augmented" NMF model, accounting for both the blocks and stripes shapes in the autosimilarity by explicitly modeling these shapes in factors. Hence, their model explicitly exploits both criteria, even if, in their final experiments, boundaries are computed considering the blocks only (and hence homogeneity).

Serrà *et al.* [Ser+14] develop "Structural Features", which, by design, encode both repetitive and homogeneous parts. The rationale of these features is to compute the similarity between bags of instances, composed of several consecutive frames. In that sense, the similarity encodes the repetition of any sequence, which can be constant (homogeneity) or evolving (repetition). Boundaries are obtained as points of high novelty between consecutive structural features.

Finally, Grill & Schlüter [GS15b] develop a Convolutional Neural Network (CNN) which outputs estimated boundaries. This CNN is one of the few techniques which do

not compute an autosimilarity to later post-process it into boundaries, but it still uses autosimilarities as input. The network is supervised on two-level annotations, on the SALAMI dataset (presented in Section 2.3.4), and, according to the authors, using these two-levels of annotations is beneficial to the performance. Practically, the network uses Log Mel spectrograms as inputs, in conjunction with autosimilarity matrices.

The boundary estimation being the result of the nonlinear function learned in the neural network, it is impossible to assume which criterion is covered by the network in its estimation. Nonetheless, Maezawa [Mae19] combined the CNN-based estimation with regularity and timbre-homogeneity constraints, and explicitly mitigate them in a combination of local optimization schemes.

Regularity

The regularity criterion assumes that segments in a song are generally lasting the same size [SBV16], or multiples of that particular size [LS06; SG16], in general computed in beats or bars (32 beats (*i.e.* 8 bars) according to Sargent *et al.* [SBV16], 4 bars and multiples for Levy & Sandler [LS06]). Hence, this criterion aims at favoring some particular sizes for segments, and is generally implemented as a constraint [SBV16].

As an example of algorithm using only the regularity criterion, Serrà *et al.* [Ser+14] compares their structural features algorithm with a regular baseline, estimating boundaries as equally spaced in each song, regardless of any other musical aspect. Experimentally, the baseline is outperformed by every other algorithm, showcasing the need for considering musical attributes in the decision-making process. We will see in Section 2.3.4 that the datasets used for evaluation on this thesis present regularity in their annotations, motivating the use of regularity constraints.

2.3.2 Labelling of Segments

In addition to the boundary retrieval task, MSA is comprised of the subtask of labelling the segments. Some of the aforementioned algorithms estimate boundaries and labels in a same framework, for instance (but not restricted to) [LS08; WB10; NJ13].

Additional works focus on estimating the labels of segments after the segmentation stage, *i.e.* once boundaries have been estimated, and hence consist of evaluating the similarity between entire sections. This can be achieved for instance with clustering techniques, such as NMF [KS10] or k-means [NB14].

We do not present much further this subtask, as this thesis focuses on the boundary retrieval subtask, but an interested reader can refer to [PMK10; Nie+20].

2.3.3 Feature Representation in MSA

The consensus in MSA is that the choice of the features impacts the type of segments considered [SC13; VOM21]. Early works were focusing on both MFCC and chromagrams, depending on the favored criterion: timbre information is considered stable across segments, hence generally used in homogeneity-based criteria such as [Foo00], and the repetition is in general favored by chromagrams and harmony, such as in [Got03]. Some work combines different features, such as [TM19], which fusions the outputs of several spectral clustering results, each computed on different features.

In recent work, Nieto & Bello [NB16] found that the Constant-Q Transform, a compact spectral representation [BP92], provided better results than MFCC or chromagrams for most of the State-of-the-Art algorithms. Nieto *et al.* extended this conclusion to Mel spectrograms [Nie+20], which is also a compact spectral representation.

The best-performing model to date [GS15b] also uses Mel spectrograms (in particular, Log Mel spectrograms). Still, we consider in this thesis that no set of features clearly stands out for structural segmentation and, as a consequence, we instead study experimentally the impact of the features on the segmentation estimations. Particularly, we compare Mel/Log Mel/NNLM spectrograms, MFCC and chromagrams, presented in Section 2.2.4, whose practical details are presented in Appendix A.1.

2.3.4 Metrics and Datasets

Evaluation baselines, in particular benchmark datasets and standard metrics, are important to quantitatively evaluate and compare different algorithms. Thus, this section presents the evaluation criteria used throughout this thesis to compare the State-of-the-Art algorithms with the methods developed in this work.

Metrics

Because this thesis focuses on boundary retrieval, we consider the Hit-Rate metrics, which compares a set of estimated boundaries with a set of annotations, by intersecting them with respect to a tolerance t [OH05; Tur+07]. Additional metrics, not considered in this thesis, are presented in [Luk08].

In practice, given two sets of boundaries Z^e and Z^a, respectively the sets of estimated and annotated boundaries, an estimated boundary $\zeta_i^e \in Z^e$ is considered to be correct if it is close enough to an annotated boundary $\zeta_j^a \in Z^a$ ("close enough" meaning that the gap is smaller than the tolerance t), *i.e.* if $\exists\, \zeta_j^a \in Z^a \,/\, |\zeta_i^e - \zeta_j^a| \leq t$. Each estimated boundary can be coupled with a maximum of one annotated boundary, and *vice versa*.

Denoting as C_t the set of correct boundaries subject to the tolerance t, C_t contains at most as many elements as the annotations or the estimations, *i.e.* $0 \leq |C_t| \leq \min(|Z^e|, |Z^a|)$. In case of perfect concordance between Z^e and Z^a, $C_t = Z^e = Z^a$. In practice, the concordance of C_t with Z^e and Z^a is evaluated by the precision P_t, recall R_t and F-measure F_t, defined as:

— $P_t = \frac{|C_t|}{|Z^e|}$, *i.e.* the proportion of accurately estimated boundaries among the total number of estimated boundaries. Hence, a high precision indicates that a large part of the estimated boundaries is correct. It may indicate some under-segmentation: the fewer boundaries are estimated, the more likely they are to be all correct, the degenerate case being the estimation of a unique boundary.

— $R_t = \frac{|C_t|}{|Z^a|}$, *i.e.* the proportion of accurately estimated boundaries among the total number of annotated boundaries. Hence, a high recall indicates that a large part of the annotated boundaries is accurately retrieved in the estimation. It may indicate some over-segmentation: the more boundaries are estimated, the more likely some of them are to match the annotations, the degenerate case being the estimation of boundaries at every instant (or spaced exactly equal to the tolerance, as in [Ser+14]).

— $F_t = \frac{2P_tR_t}{P_t+R_t}$ is the harmonic average of both aforementioned measures. The harmonic average is less sensible to large values than the arithmetic (standard) average, and is conversely more strongly penalized by low values. Hence, a high F-measure requires both a high recall and a high precision, which may ensure a trade-off between obtaining accurate boundaries and spanning the annotations.

In structural segmentation, common conventions for the tolerance values are 0.5s [Tur+07] and 3s [OH05]. The 3 seconds tolerance, citing Ong & Herrera [OH05], is justified as being equal to "approximately 1 bar for a song of quadruple meter [NB: 4 beats per bar, *e.g.* $\frac{4}{4}$ metric] with 80 bpm in tempo", while the 0.5 second tolerance is at an order of magnitude corresponding to the beat.

In this thesis, we use both tolerance values, leading to 6 metrics $P_{0.5}$, $R_{0.5}$, $F_{0.5}$ and P_3, R_3, F_3. In practice, these metrics are computed using the *mir_eval* toolbox [Raf+14].

Note that the tolerance considers the absolute value of the difference between the estimation and the annotation. Hence, it can equivalently be considered that an estimation is correct if it is contained in a window of size $2t$ centered on the annotation, in our case windows of 1s and 6s respectively.

Datasets

Several standard datasets, with structural annotations, are available for research purposes nowadays [Got+02; Smi+11; Nie+19; Mau+09]. Still, while annotations are easily shareable, copyright infringement issues can complicate the distribution of the actual audio files.

In that sense, Goto *et al.* [Got+02] designed the RWC dataset, a copyright-cleared dataset, designed by and for researchers in MIR. The dataset is composed of four subsets: the *Pop*, *Classical Music* and *Jazz* datasets, which contain songs according to the eponymous musical styles, and the *Royalty-Free Music* dataset, which contain public-domain traditional pop music. This thesis focuses on the Pop dataset, hereafter designed as the **RWC Pop** dataset.

The RWC Pop dataset is composed of 100 songs, all originally written, composed and recorded by professional musicians and sound engineers, with the aim of achieving sound quality equivalent to that of commercially distributed music. Two sets of structural annotations are available for this dataset: AIST [Got+06] and MIREX10 [Bim+10].

AIST annotations were designed by the original authors of the dataset, and were the first annotations publicly available. Bimbot *et al.* later produced the MIREX10 annotations [11] precisely following the methodological guideline described in [Bim+10]. The authors aimed at producing annotations which are consistent across annotators and datasets, in order to reduce the inter-annotator ambiguity which exists in structural evaluation. In general, in the MIREX contest [12], the MIREX10 set of annotations leads to significantly better segmentation scores compared to the AIST set of annotations.

Hence, in this thesis, we only consider the MIREX10 set of annotations for experimental results. It is to be noted that the MIREX10 set of annotations does not provide label annotations, while the AIST set of annotations does. Though, we recall that labels are not evaluated in this thesis.

In this thesis, we also consider the **SALAMI** dataset [Smi+11], which, to this date,

11. The name stems from the 2010 MIREX contest, where the set of annotations was introduced.
12. *e.g.* the 2016 results: www.music-ir.org/mirex/wiki/2016:MIREX2016_Results

is the largest publicly available dataset of professionally recorded music with structural annotations. This dataset is composed of 1359 songs, labelled as belonging to popular, jazz, classical and world music styles.

This dataset is composed of annotations at two hierarchical levels: fine (corresponding to melodic motifs and phrases) and coarse (related to sections). The coarse level is the hierarchical level which best corresponds to the "flat" segmentation targeted in this thesis. In addition, 884 songs are annotated by at least two annotators, and, when faced with several annotations, we keep the closest to our estimation (*i.e.* the best value for the average between $F_{0.5}$ and F_3).

Evaluation Limits

Citing Bruderer *et al.* [BMK06], "Algorithms that segment music are often binary in nature", but "the perception of boundaries is not binary". Indeed, musical structure is highly subjective, and depends on the musical properties under consideration as well as on the musical background of the listener.

In that spirit, Nieto *et al.* proposed to adapt the F-measure to better represent the perceptual evaluation of boundaries. According to the authors, the precision metric is more relevant perceptually than the recall when evaluating estimated boundaries. In the same spirit, Wang *et al.* studied the subjectivity in annotations [WMD17], and concludes that some boundaries are more consensual than others. The authors suggest to refine the methodology and definitions of structure, in particular for annotators.

While some authors developed guidelines for refining musical structure definition, *e.g.* [Bim+10; Smi+11; Bim+12], the structure remains ambiguous in its essence [Nie+20], and more accurate systems should clearly consider ambiguity, for instance by differentiating the structure depending on the instrument considered [SG17], by considering structure as a multi-dimensional object [PD09], or by considering different metrics to quantitatively evaluate the structure [Luk08; GGL16].

These considerations must be kept in mind when evaluating and comparing algorithms, but they are beyond the scope of this thesis, which only attempts to provide new algorithms and representations to evaluate boundaries in an audio musical signal.

Algorithms Used as State-of-the-Art

Despite the large literature on MSA, we restrict the choice of State-of-the-Art algorithms to only five algorithms. Firstly, we have chosen four unsupervised algorithms,

namely Foote's kernel [Foo00], Convex NMF from Nieto & Jehan [NJ13], Spectral Clustering from McFee & Ellis [ME14a], and the Structural Features from Serrà *et al.* [Ser+14].

All these algorithms are developed in the *MSAF* toolbox [NB16]. Foote's kernel is motivated as State-of-the-Art algorithm because, despite its simplicity, it is still one of the best-performing algorithms nowadays, and is still used in many recent articles for comparison. Comparison with Convex NMF is motivated by its conceptual proximity to some factorization and compression schemes presented in this thesis (particularity NTD and NMF, respectively Sections 4.2.2 and 5.3.1). Finally, the Spectral Clustering and the Structural Features are also considered because they are the best-performing unsupervised methods.

As global State-of-the-Art though, we consider the CNN of Grill & Schlüter [GS15b], which is the best-performing algorithm, and keep the results publicly available from the 2015 MIREX contest. However, this approach is a supervised method, as it requires a larger number of training data. In particular, this method is supervised on a subset of the SALAMI dataset, the remaining part of the dataset being used as a test dataset (stemming from the MIREX contest). Details about the learning/test repartition of the SALAMI dataset are presented in Appendix A.2.

2.4 Focus on Barwise Music Processing

As previously mentioned, this thesis studies structure at the barscale, *i.e.* it considers that boundaries occur precisely between bars, on downbeats, and that sections are composed of several whole bars. This section aims at detailing this viewpoint both in terms of motivations and practical considerations.

2.4.1 Motivations

In previous work on MSA, features were either computed with a fixed hop length, typically between 0.1s and 1s [PMK10], or (in the more recent works), aligned on beats [Nie+20]. The beat alignment is musically-relevant because it aligns the features and the estimations with respect to a time segmentation consistent with music performance.

In this thesis, we consider that the barscale is more relevant than the beat scale to study structural segmentation of Pop and Western modern music (which are the styles of both RWC Pop and SALAMI). In our opinion, bars are well suited to express patterns

and sections in Western modern music.

Indeed, in the conventions of Western musical notations (as presented in Section 2.2.2), musical notes are expressed relatively to beats, and beats are combined to form bars. Bars finally segment the musical scores (with vertical lines), and repetition occurs generally between different bars (which is particularly visible by the use of repeat bars, or symbols as "Dal segno", "Da Capo", etc).

In addition, the intuition that musical sections start and end on downbeats is experimentally confirmed by works such as [MND09; Fue+19], where the use of structural information improves the estimation of downbeats. The direct drawback is the need for a powerful tool to estimate bars.

In this thesis, we use the *madmom* toolbox [Böc+16]. In details, the *madmom* toolbox uses a neural network to perform the estimation, described in [BKW16]. In the 2016 MIREX contest [13], which was the last edition of the contest comparing downbeat estimation algorithms, this neural network obtained the best performance, and can hence be considered as a State-of-the-Art algorithm for the task.

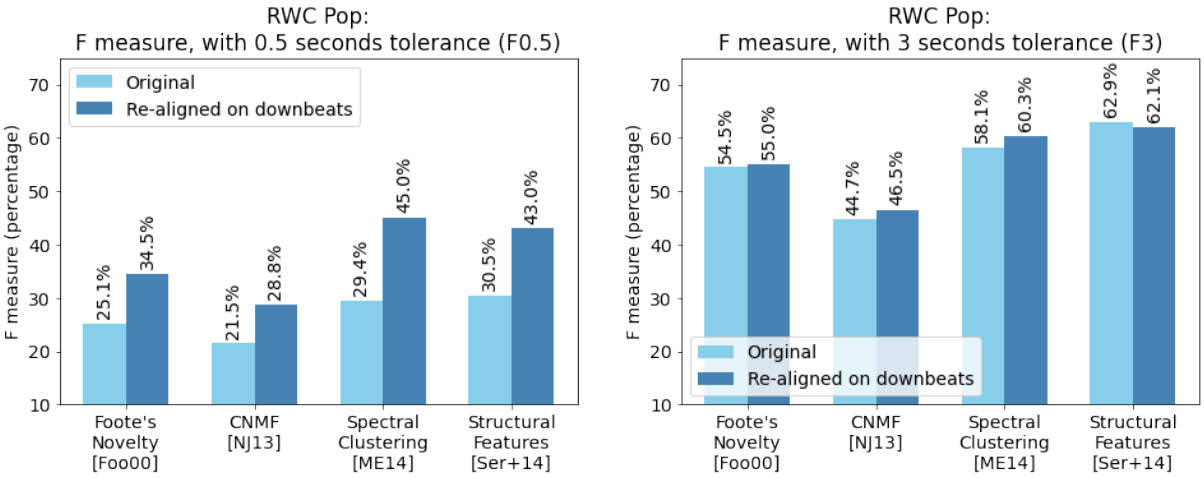
To support the idea of barwise estimation of structure, we estimate boundaries with the four unsupervised State-of-the-Art algorithms, computed with the *MSAF* toolbox [NB16], and post-process the segmentation outputs by aligning each boundary with the closest downbeat. A comparison of the segmentation scores with and without the alignment is presented in Figure 2.14 for both RWC Pop and SALAMI datasets. In these algorithms, the original boundaries are aligned with beats, thus, we are comparing segmentation results either beatwise or barwise aligned. These results were already presented in [Mar+20] for some of the algorithms, and on the RWC Pop dataset only.

Results show that aligning estimated boundaries on downbeats results in a strong increase in performance for $F_{0.5}$, and to comparable results for F_3, on both datasets. Hence, aligned on downbeats, estimations are more accurate, but the F_3 metric is not significantly impacted by this alignment. We recall that the 3 seconds tolerance was initially motivated as corresponding approximately to one bar [OH05], which is consistent with the lack of significant difference in the results. Thus, in the remainder of this thesis, results of these four State-of-the-Art algorithms are given aligned on downbeats.

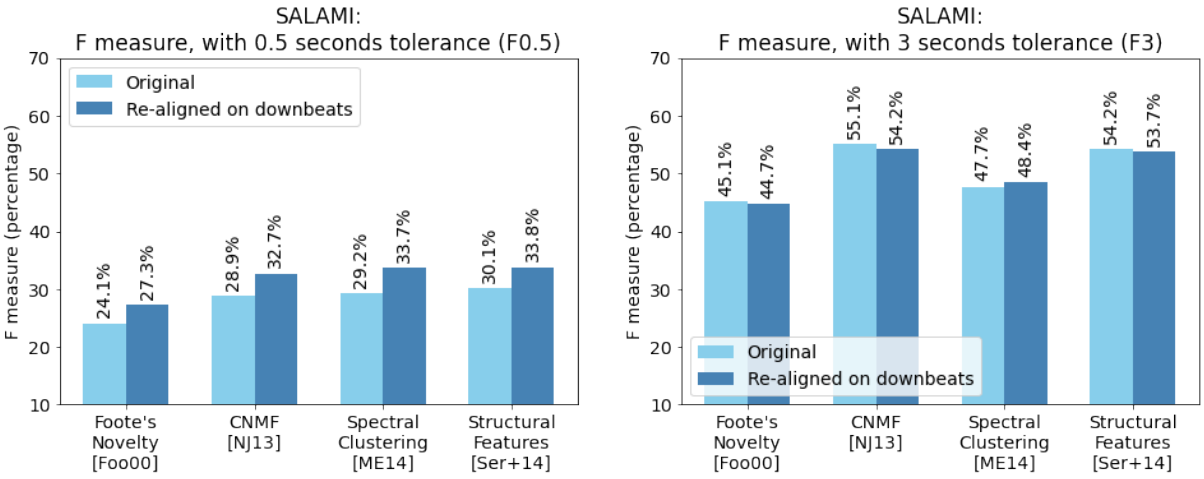
For the CNN [GS15b], global State-of-the-Art, we used results obtained at the 2015 MIREX contest [14] (hence, not aligned barwise), as the code and the experimental frame-

13. www.music-ir.org/mirex/wiki/2016:Audio_Downbeat_Estimation_Results
14. nema.lis.illinois.edu/nema_out/mirex2015/results/struct/mrx10_1/

work are not publicly available.



(a) RWC Pop.



(b) SALAMI (test subset).

Figure 2.14 – Segmentation results of State-of-the-art algorithms on the RWC Pop and the SALAMI datasets, for beatwise (original) and barwise aligned boundaries.

2.4.2 Barwise Processing of Inputs

The remainder of this thesis studies structure at the barscale, motivated by the aforementioned arguments and experimental results reported in Figure 2.14. Nonetheless, instead of post-processing the estimations, we rather pre-process the input features.

In this setting, each frame of information holds the context of exactly one bar, *i.e.* barwise sampled features. Indeed, following the idea that music is organized barwise, in

different patterns, the rationale of the methods studied in this thesis is to catch these barwise patterns. In addition, barwise alignment allows to cope with tempo variations in songs, and catch patterns even if performed with some temporal inaccuracies.

The direct drawback of this alignment is that it hinders all differences in tempo, assuming that tempo should be steady across the song. In some musical contexts, variations of tempo are an important part of the musical content, and some structural boundaries can be based solely on changes of tempo, as discussed in [VOM21]. We do not propose a strategy to counteract this effect, and assume that, in our case study (*i.e.* Pop and modern Western music), the benefits of lowering imperfections in performance and studying patterns at the barscale are greater than trying to detect boundaries solely based on tempo differences.

Practically speaking, the feature representation is re-sampled in barwise spectrograms. Firstly, spectrograms are computed (in any feature representation) with a low hop length of 32 frames. Denoting as F the dimension of the feature representation, presented in Section 2.2.4, and Ω the number of frames, this results in an oversampled spectrogram of size $F \times \Omega$. Then, downbeats are estimated with the *madmom* toolbox. This allows us to split the original spectrogram in B barwise spectrograms (B being the number of bars in this song) each containing Ω_B frames. As bars can be of different lengths (because of differences in tempo or inaccuracies in the performance), different bars can contain a different number Ω_B of frames.

Thus, we define a **subdivision** parameter S, which is the desired number of time frames in each bar. Hence, barwise sampling transforms the spectrogram of size $F \times \Omega$, in a "barwise sampled" spectrogram of size $F \times BS$ (with $BS < \Omega$), and this process is schematized in Figure 2.15.

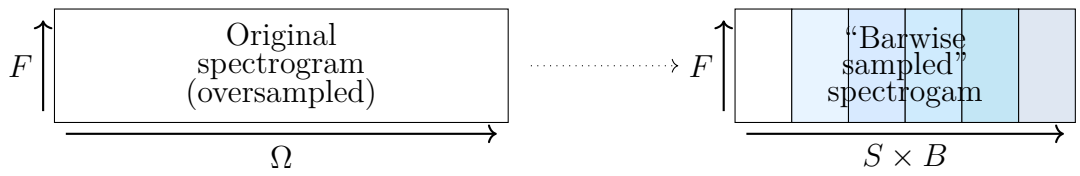


Figure 2.15 – Barwise sampled spectrogram, *i.e.* a representation where each bar contains the same number S of time frames.

Practically, starting from the subdivision S, and from indexes ω_1 and ω_2, respectively the indexes of the closest frames to the downbeats starting and ending the bar, barwise sampling consists of selecting all frames $\left\{\omega_1 + \lfloor \frac{k(\omega_2-\omega_1)}{S} + \frac{1}{2} \rfloor, 0 \leq k < S, k \in \mathbb{N}\right\}$, *i.e.*

equally-spaced frames in the bar to fit the chosen subdivision, as presented for a particular bar in Figure 2.16 [15].

A different strategy [SG18], not considered in this thesis, consists of taking the length of the largest bar as the desired dimension, and zero-pad all smaller barwise spectrograms.

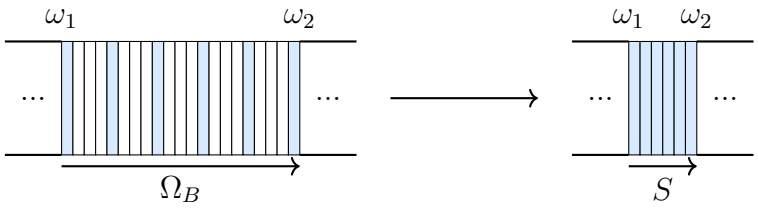


Figure 2.16 – Barwise sampling: zoom on a bar. Here, the bar corresponding to the original feature representation is composed of $\Omega_B = 21$ frames, and the barwise sampled bar of $S = 6$ frames.

By vectorizing each barwise spectrogram (of size $F \times S$) in the "barwise sampled" spectrogram, and concatenating them along the bar dimension, we introduce the **Barwise TF** representation, consisting of a matrix of size $B \times FS$. Note that we choose to vectorize the time-frequency features, thus discarding the dependencies between the time and frequency dimensions. The aforementioned process is described in Figure 2.17.

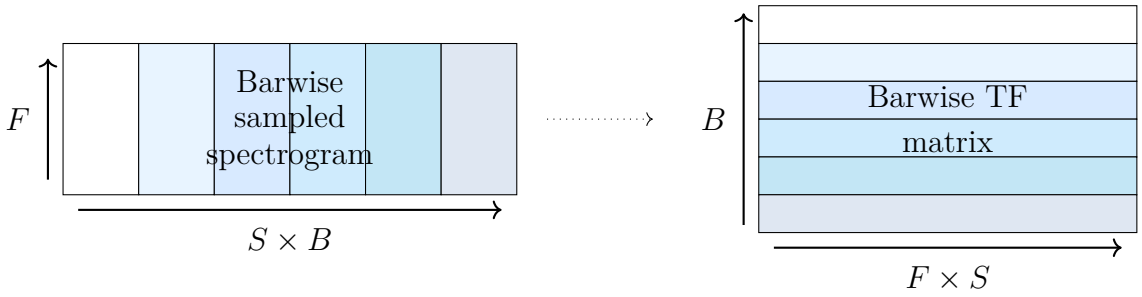


Figure 2.17 – Barwise TF matrix.

In early tests, we compared three values for S: $96, 128$ and 192, empirically chosen as good fits for time signatures multiple of 4 (as they are all multiple of 4), typically $\frac{4}{4}$, but also of 3 (for 96 and 192), typically $\frac{3}{4}$, as both $\frac{4}{4}$ and $\frac{3}{4}$ time signatures seem empirically to be the most common time signatures in Pop Music. In addition, we wanted these subdivisions to be large enough to catch important musical information in

15. Other techniques could be applied to reduce the number of frames (for example averaging the content of all frames instead of choosing one), but we did not pursue that lead.

the bars [16]. Experiments showed no significant differences between these three values, so we fixed it to 96, the lowest value, to gain in calculus complexity.

2.4.3 Dataset Analysis

Of particular importance for this thesis is the study of the impact of barwise processing on the datasets. Firstly, to study the impact on the annotations, Table 2.1 presents the structural segmentation metrics when comparing the annotations aligned on the closest downbeats with the original annotations. Comparing the annotations re-aligned on downbeats with their original values provides the upper limit of metrics when using barwise aligned features.

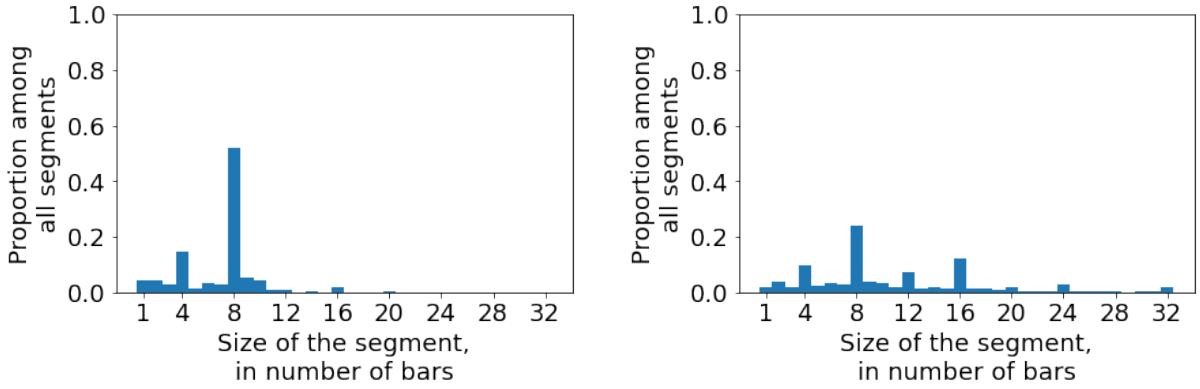
The performance of the barwise-aligned annotations are way higher than those of the current State-of-the-Art (see Figure 2.14), which indicates that barwise processing may not be a problem as long as the associated methods gain in performance in the estimation of boundaries. Still, while the annotations of RWC Pop seem to be precisely located on downbeats, the SALAMI annotations obtain lower results for the metrics with the 0.5s tolerance, indicating either wrong downbeats estimation or annotations which are less systematically synchronized on the downbeats.

Dataset		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
RWC Pop		96.46%	96.21%	96.33%	100%	99.73%	99.86%
SALAMI (test subset)	Annotation 1	82.89%	82.85%	82.87%	99.95%	99.91%	99.93%
	Annotation 2	81.94%	81.89%	81.91%	99.96%	99.88%	99.92%

Table 2.1 – Metrics when aligning the references on the downbeats (compared to the original annotations). The SALAMI dataset is restricted to the test subset, as detailed in Appendix A.2, and the two sets of annotations are presented.

From these barwise annotations, Figure 2.18 presents the distribution of the sizes of segments, in terms of number of bars in the annotations. This distribution is important regarding the criterion of regularity, as the majority of segments in both datasets are of size 8 bars, and the remainder are mostly of sizes 4, 12 and 16 bars (especially in SALAMI). Hence, methods employed on these datasets should favor these sizes.

16. In a $\frac{4}{4}$ metric, a subdivision of 96 corresponds to 24 frames per beat.



(a) RWC Pop.

(b) SALAMI (test subset).

Figure 2.18 – Distribution of segments sizes, in terms of number of bars, in the annotations. The SALAMI dataset is restricted to the test subset, as detailed in Appendix A.2, and both sets of annotations are mixed here.

2.5 Conclusions

This section presented the important tools to process music computationally, such as standard music theory concepts and important features to describe music.

Furthermore, this section has reviewed the literature in structural segmentation, a subtask of the larger Music Structure Analysis task, typically divided in four criteria: homogeneity, novelty, repetition and regularity. Algorithms may focus on one or more of these criteria to estimate the different sections in a song.

Finally, we introduced the notion of barwise aligned features, along with the Barwise TF matrix representation, which consists of processing the original spectrogram in barwise representations. The next chapter introduces the use of this representation for structural segmentation, along with a new segmentation algorithm, the CBM algorithm.

Chapter 3

Convolutive "Block-Matching" Segmentation Algorithm

Synopsis

This chapter details the segmentation algorithm used throughout this thesis. This algorithm is called the CBM algorithm, standing for *Convolutive Block-Matching*.

3.1 Introduction

This chapter introduces a segmentation algorithm, *i.e.* an algorithm which computes a set of boundaries Z^e from a feature representation of the song. In practice, this algorithm works at the barscale, *i.e.* it takes a barwise representation of the song as input (in this chapter, the Barwise TF matrix). This algorithm is called Convolutive "Block-Matching" (CBM) segmentation algorithm, and focuses on the homogeneity/novelty and regularity criteria, presented in the Section 2.3.

The first part of this chapter is dedicated to the formal definition of barwise autosimilarity matrices, representing the similarity between all pairs of bars in a song. These autosimilarities are computed according to different similarity functions, namely Cosine, Covariance and RBF. The principle of the CBM algorithm, along with the details of important parameters in this algorithm (namely the convolution kernels and penalty functions) are presented in a second part. Finally, this chapter presents the segmentation performance of this algorithm on both the RWC Pop and SALAMI datasets in a third part.

The contributions reported in this section are twofold:

— Algorithmic: While the use of dynamic programming for estimating the structure in a song was already presented in previous works [Jen06; SBV16], the current CBM algorithm extends both algorithms by redesigning the score computation for a segment. In that sense, the CBM algorithm is more flexible, and the estimation of segments can be adapted to specific hypotheses on their definition.
In addition, this chapter extends the computation of autosimilarity matrices to similarity functions not yet explored in the audio segmentation literature (in particular, the Radial Basis Function).

— Experimental: The current version of the CBM algorithm, specifically tuned for barwise processing of music, obtains experimental results achieving a level of performance close to those of current State-of-the-Art methods.

This segmentation algorithm has been used in three publications [Mar+20; Mar+22; MCB22c], and a dedicated publication is currently under review [MCB22d] [1]. The CBM algorithm is implemented in the open-source *as_seg* toolbox [MCB22a].

1. An international journal publication, fully dedicated to the CBM algorithm, is currently under preparation.

3.2 Autosimilarity Matrix

Given a matrix $X \in \mathbb{R}^{B\times M}$, an autosimilarity of X is defined as a matrix $A(X) \in \mathbb{R}^{B\times B}$ where each coefficient (i, j) represents the similarity between vectors X_i and X_j. The similarity between two vectors is subject to a similarity function (the dot product for instance), and, as a consequence, different autosimilarity matrices can be constructed, depending on the choice of the similarity function. The main diagonal in an autosimilarity matrix represents the self-similarity of each vector, and is in general (and in this thesis in particular) normalized to one. This thesis studies three different similarity functions (and, subsequently, three different autosimilarity matrices), namely the Cosine, Covariance and RBF similarity functions.

In this chapter, an autosimilarity matrix is computed on the Barwise TF matrix, *i.e.* $X \in \mathbb{R}^{B\times SF}$, which consists of the time-frequency representations for each bar. Hence, each coefficient in the autosimilarity represents the feature-wise similarity for a pair of bars.

3.2.1 Cosine Autosimilarity Matrix

The Cosine similarity function computes the normalized dot products between two vectors, and leads to the Cosine autosimilarity matrix, denoted as $A_{cos}(X)$. Practically, denoting as $\tilde{X}$ the row-wise l_2-normalized version of X (*i.e.* the matrix X where each row has been divided by its l_2-norm), the Cosine autosimilarity matrix is defined as $A_{cos}(X) = \tilde{X}\tilde{X}^\intercal$, or, elementwise, for $1 \leq i, j \leq B$:

$$A_{cos}(X)_{ij} = \frac{\langle X_i, X_j\rangle}{\|X_i\|_2\|X_j\|_2} = \sum_{k=1}^{TF} \tilde{X}_{ik}\tilde{X}_{jk}. \tag{3.1}$$

This similarity function is called the "Cosine" similarity because the normalized dot product is equal to the cosine of the angle between both vectors [2].

The normalized dot product is related to the Euclidean distance $d(X_i, X_j)$ (in the Euclidean space associated with the TF dimensions) by [3] $\frac{d(X_i,X_j)^2}{2} = 1 - \langle X_i, X_j\rangle$. Hence, the larger is the dot product, the smaller is the distance. This attests that the Cosine autosimilarity matrix represents similarity between pairs of vectors.

2. Trivially recalling that $\langle x, y\rangle = \|x\|_2\|y\|_2 \cos(\widehat{xy})$.

3. This is straightforward as $d(x, y)^2 = \langle x, x\rangle + \langle y, y\rangle - 2\langle x, y\rangle$, recalling that $\langle x, x\rangle = \langle y, y\rangle = 1$ in our case.

3.2.2 Covariance Autosimilarity Matrix

The Covariance of two random variables a and b (seen as row vectors), is defined as $cov(a, b) = \mathbb{E}\Big[(a - \mathbb{E}[a])(b - \mathbb{E}[b])^\intercal\Big]$. In our discrete and non-probabilistic context, we define the Covariance similarity function for 2 bars X_i and X_j as $cov(X_i, X_j) = (X_i - \bar{x})(X_j - \bar{x})^\intercal$, denoting as $\bar{x} \in \mathbb{R}^{TF}$ the average of all bars in the song.

The barwise Covariance similarity function defines the Covariance autosimilarity matrix $A_{cov}(X)$ as:

$$A_{cov}(X)_{ij} = \frac{\langle X_i - \bar{x}, X_j - \bar{x} \rangle}{\|X_i - \bar{x}\|_2 \|X_j - \bar{x}\|_2}. \tag{3.2}$$

In other words, the Covariance matrix is exactly the Cosine autosimilarity matrix of the centered matrix $X - \mathbb{1}_B \bar{x}$, *i.e.* $A_{cov}(X) = A_{cos}(X - \mathbb{1}_B \bar{x})$.

3.2.3 RBF Autosimilarity Matrix

Kernel functions are symmetric positive definite or semi-definite functions. In machine learning, kernel functions are generally used to represent data in a high-dimensional space (sometimes infinite), enabling a nonlinear processing of data with linear methods (for instance, nonlinear classification with Support Vector Machines, SVM). In particular, the "kernel-trick" allows to study relations in the data in a high-dimensional space without directly embedding the data in that space, which would have been inefficient, or even intractable, in practice.

In particular, the Radial Basis Function (RBF) kernel is a kernel function defined as $\mathrm{RBF}(X_i, X_j) = \exp(-\gamma \|X_i - X_j\|_2^2)$, with γ a user-defined parameter. The RBF can be used as a similarity function between two bars X_i and X_j, hence defining the RBF autosimilarity matrix $A_{RBF}(X)$ as:

$$A_{RBF}(X)_{ij} = \mathrm{RBF}(\tilde{X}_i, \tilde{X}_j) = \exp\left(-\gamma \left\| \frac{X_i}{\|X_i\|_2} - \frac{X_j}{\|X_j\|_2} \right\|_2^2\right). \tag{3.3}$$

Bars are normalized by their l_2 norm in the computation of A_{RBF}, in order to limit the impact of variations of power between the different bars. The self-similarity of a bar is equal to $e^0 = 1$, hence the RBF autosimilarity matrix does not require normalization to result in self-similarities equal to one.

Parameter γ is set relatively to the standard deviation of the pairwise Euclidean distances of all bars in the original matrix (self-distances excluded), to adapt the shape of the

exponential function to the relative distribution of distances in this song. Hence, denoting as $\sigma = \underset{1<i,j<B, i\neq j}{std} \left(\left\| \frac{X_i}{\|X_i\|_2} - \frac{X_j}{\|X_j\|_2} \right\|_2^2 \right)$, we set $\gamma = \frac{1}{2\sigma}$.

These three autosimilarities are presented in Figure 3.1, on the Barwise TF (in Log Mel spectrogram) of the song *POP01* of RWC Pop.

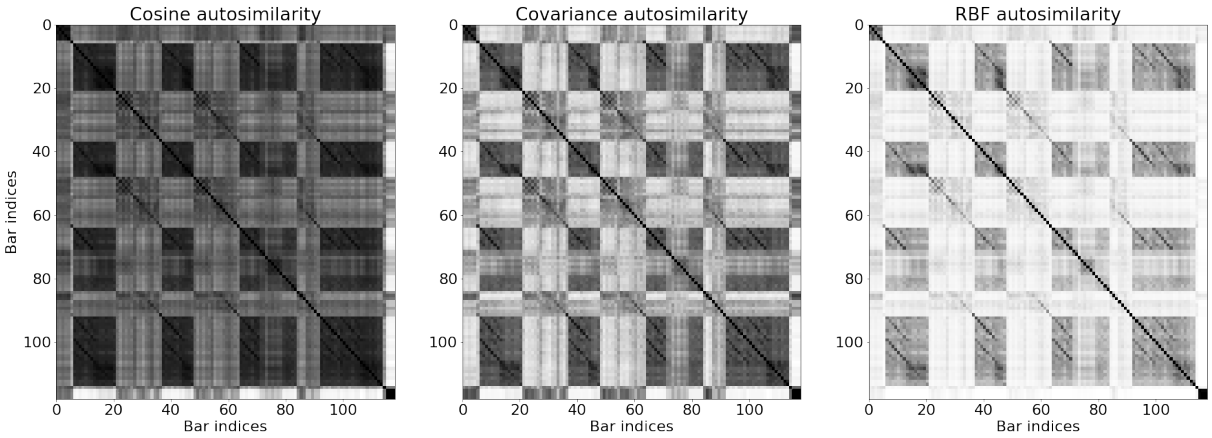


Figure 3.1 – Cosine, Covariance and RBF autosimilarities on the song *POP01* of RWC Pop, in the Log Mel feature.

3.3 Convolutive "Block-Matching" Algorithm

Starting from an autosimilarity matrix, structural segmentation is obtained by means of a dynamic programming algorithm, called Convolutive "Block-Matching" segmentation algorithm (CBM). This algorithm was previously introduced in [Mar+20; MCB22c]. This section aims at presenting this algorithm in details, along with a study of important parameter settings.

The CBM algorithm estimates structural segmentation based on the homogeneity/novelty and regularity criteria. As a reminder, the homogeneity/novelty criteria assume that sections can be defined as highly similar regions (*i.e.* regions with little inner variations), and that boundaries should be placed at breaking points. Regarding the regularity criterion, it assumes that boundaries should preferably be spaced evenly.

3.3.1 Structural Segmentation Solved as a Dynamic Programming Algorithm

In the CBM algorithm, segmentation is estimated by means of a dynamic programming algorithm. The Viterbi and Dynamic Time Warping (DTW) algorithms are examples of dynamic programming algorithms with a lot of applications in the Audio community. As reported in Section 2.3, several dynamic programming algorithms have already been developed for structural segmentation of music.

Of particular interest is the work of Jensen [Jen06], later extended by Sargent *et al.* [SBV16] (to account for the regularity principle). The CBM algorithm is based on the principles introduced in these articles, which are reformulated hereafter in a first part.

In a nutshell, the CBM algorithm is based on the definition of a score function $\gamma()$ applied on segments, with the overall segmentation of the song resulting in the maximum total score of the segments. This defines an optimization problem, which can be solved by dynamic programming. The details of the score function are presented in a second part.

Definition of the Problem

Formally, a segmentation is defined as a set of boundaries $Z = \{\zeta_i,\, i \in [\![1, E]\!]\}$, ζ_i being located on a time-frequency frame, and E representing the number of boundaries estimated. The set of admissible segmentations is denoted as Θ, *i.e.* $Z \in \Theta$.

Each segment S_i is composed of the time-frequency frames between two consecutive boundaries, *i.e.* $S_i = [\![\zeta_i, \zeta_{i+1}[\![$. The second bound is exclusive as it represents the start of the consecutive segment S_{i+1}. Boundary ζ_i is called the **antecedent** of boundary ζ_{i+1}.

In our paradigm, each boundary is located on a bar, *i.e.* $\forall i,\, \zeta_i \in [\![1, B+1]\!]$, with B the number of bars in this song [4]. The first boundary is the start of the song, *i.e.* $\zeta_1 = 1$, and the last boundary is the end of the last bar in the song, *i.e.* $\zeta_E = B + 1$. Without prior knowledge, any bar can constitute a boundary.

In this formulation, each set of boundaries consists of selecting E bars as boundaries among $B+1$. As a consequence, there exists $\binom{B+1}{E}$ different sets of boundaries composed of exactly E boundaries, and, more generally, at most $\sum_{k=0}^{B+1} \binom{B+1}{k} = 2^{B+1}$ segmentations for each song [5]. Hence, the segmentation problem admits a finite number of solutions,

4. As the song contains B bars, $B+1$ represents the end of the last bar, *i.e.* the last downbeat of the song.

5. In fact, as each set of boundaries must contain the first and last downbeats of the song, at most

which can theoretically be solved in a combinatorial way. In practice though, evaluating all possible segmentations leads to an algorithm of exponential complexity $\mathcal{O}(2^B)$, considered intractable in practice.

Dynamic Programming

The segmentation problem is approached by both Jensen and Sargent *et al.* [Jen06; SBV16] as an optimization problem. In particular, by associating a score $\gamma(S)$ to each potential segment S, the optimal segmentation Z^* is the segmentation maximizing [6] the sum of all its segment scores [7]:

$$Z^* = \underset{Z \in \Theta}{\arg\max} \sum_{i=1}^{E-1} \gamma([\![\zeta_i, \zeta_{i+1}[\![). \tag{3.4}$$

The problem can be solved using a dynamic programming algorithm. Dynamic programming is an algorithmic method, which can be employed to solve particular optimization problems [Cor+09, Chap. 9].

Dynamic Programming

The principle of dynamic programming is to solve a combinatorial optimization problem by dividing it into several independent subproblems. The independent subproblems are formulated in a recursive manner, and their solutions can be stitched together to form a solution to the original problem.

Notice that in the current formulation of the segmentation problem, defined in Equation 3.4, each potential segment is evaluated independently, via its score, and is never compared with the others. In other terms, repetitions of the same section are not considered, while they could inform on the overall structure, typically considering the repetition criterion.

2^{B-1} sets of boundaries can be obtained.

6. In details, both Jensen and Sargent *et al.* [Jen06; SBV16] introduced the optimal segmentation as the minimum of a cost function, when it is rather defined here as a maximum. It actually depends on the way of conceiving the score function $\gamma()$, and, in particular, by defining a cost function equal to the opposite of the score function $\gamma()$, both problems are equivalent.

7. Notation "arg max" means finding the elements maximizing the score.

Thus, the segmentation problem defined in Equation 3.4 is a relaxation of the general segmentation problem. This relaxation is considered because it allows us to use principles of dynamic programming, by evaluating the score of each segment independently, as independent subproblems. In particular, this relaxed problem is said to exhibit "optimal substructure", defined by Cormen *et al.* as the following property: "optimal solutions to a problem incorporate optimal solutions to related subproblems, which we may solve independently" [Cor+09].

Longest-Path on a Directed Acyclic Graph

Following the formulation of Jensen [Jen06], the segmentation problem is reframed into the problem of finding the longest path on a Directed Acyclic Graph (DAG) [8].

Directed Acyclic Graph (DAG)

A DAG is a **graph**, composed of vertices and edges. Edges between vertices are **directed**, meaning that an edge is directed from a vertex to one another (for instance $u \to v$). The graph is **acyclic**, meaning than no loop can be formed by following consecutive edges (for instance, $u \to v \to w \to u$ is impossible).

By considering each possible boundary ζ_i (in the standard case, each bar in the song) as a vertex, and each segment $S_i = [\![\zeta_i, \zeta_{i+1}[\![$ as an edge, a segmentation can be reinterpreted as a path in a DAG. By assigning the score of each segment as the length of the associated edge, the optimal segmentation of Equation 3.4 is exactly the problem of finding the longest path in the graph.

In addition, the graph is topologically ordered (due to the chronological order of bars in the song), and is both composed of a single vertex as origin (the first bar of the song) and a single final vertex (the last bar). This problem is presented and solved in [Cor+09, Chap. 24], for both shortest and longest paths, as the "Single-Source Shortest Paths in Directed Acyclic Graphs" problem.

The rationale of the solution algorithm is that the optimal segmentation until any given bar b_k (any vertex) can be found exactly by recursively evaluating the optimal segmentations until each antecedent of b_k, *i.e.* (without any constraint) all bars $b_l < b_k$, and

8. Precisely, shortest path in the minimization problem, longest path in the maximization problem.

the score of the segments $[\![b_l, b_k[\![$. Formally, denoting as $Z^*_{[1:b_k]}$ the optimal segmentation until bar b_k, the algorithm consists of:

1. Computing [9] $\{\gamma(Z^*_{[1:b_l]}), \forall b_l < b_k\}$, *i.e.* the longest paths connecting every antecedent,
2. Computing $\{\gamma([\![b_l, b_k[\![)\}$, *i.e.* the length of the edge between vertices b_l and b_k,
3. Finding the best antecedent of b_k, denoted as $\zeta^*_{b_k-1}$, with the following equation:

$$\zeta^*_{b_k-1} = \arg\max_{b_l}(\gamma(Z^*_{[1:b_l]}) + \gamma([\![b_l, b_k[\![)). \tag{3.5}$$

Finally, at the last iteration, the algorithm computes the best antecedent for $B+1$, *i.e.* the last downbeat of the song. Then, recursively, the algorithm is able to track back the best antecedent of this antecedent, and so on and so forth until the first bar of the song, thus leading to the optimal segmentation. A graph visualization for a 4 bars example is presented in Figure 3.2.

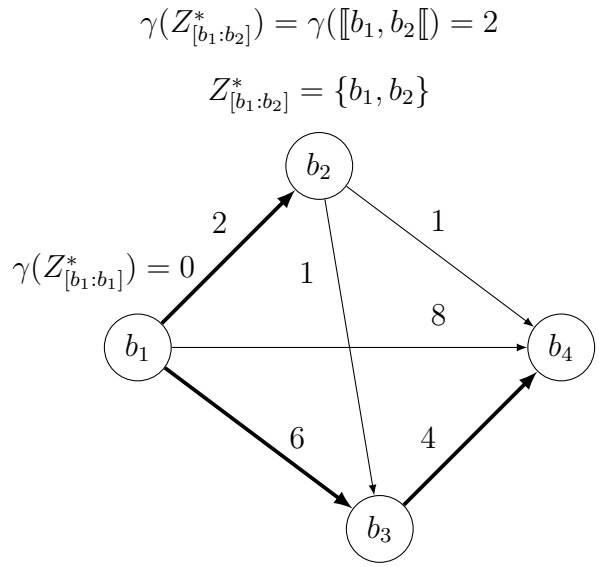


Figure 3.2 – Longest path resolution for a DAG with 4 bars. The lengths of each edge corresponds to the score of the segment between the associated bars, *i.e.* $\gamma([\![b_i, b_j[\![)$.

9. In practice, as presented in Algorithm 1, longest paths are stored at first computation, hence bringing the complexity of this operation to the evaluation of an array.

Algorithm 1: Dynamic programming algorithm, computing the optimal segmentation given a score function $\gamma()$.

Input: Bars $\{b_k \in [\![1, B]\!]\}$, score function $\gamma()$
Output: Optimal segmentation $Z^* = \{\zeta_i\}$
$Z^* = \{1, B\}$

$A^* = [\,]$
`/* Array storing the optimal antecedents for every bar (empty at initialization). */`

$\Gamma^* = [0]$
`/* Array storing the optimal segmentation until each bar (set to` $\Gamma^*[1] = 0$ `at initialization). */`
for $b_k = 2, ..., B + 1$ **do**
  $\zeta = b_k$
  `/* Considering bar` b_k `as current boundary` ζ`. */`

  $\zeta^*_{-1} = \underset{1 \leq \zeta_j < \zeta}{\arg\max}\ (\Gamma^*[\zeta_j] + \gamma([\![\zeta_j, \zeta[\![))$
  `/* Finding the best antecedent for the current boundary` ζ `with Equation 3.5. */`

  $A^*[\zeta] = \zeta^*_{-1}$
  `/* Storing the best antecedent for` ζ`. */`

  $\Gamma^*[\zeta] = \Gamma^*[\zeta^*_{-1}] + \gamma([\![\zeta^*_{-1}, \zeta[\![)$
  `/* Computing and storing the optimal segmentation until` ζ`. */`
end
$\zeta = B + 1$
while $A^*[\zeta] \neq 1$ **do**
  `/* Recursively tracking back all optimal antecedents, from the last to the first bar. */`

  $\zeta^*_{-1} = A^*[\zeta]$
  $Z^* = Z^* \cup \{\zeta^*_{-1}\}$
  `/* Searching for the best antecedent and adding it to the optimal segmentation. */`

  $\zeta = \zeta^*_{-1}$
  `/* Iterating the process with the current best antecedent. */`
end

The detailed algorithm, assuming that the score function $\gamma()$ is given, is detailed in Algorithm 1. Practically, the advantage of the algorithm is to be able to store in memory both the optimal antecedent for each bar and the scores of the optimal segmentation until each bar when they are computed for the first time, respectively denoted as the arrays A^* and Γ^* in Algorithm 1. Storing these two quantities in memory allows to gain in complexity by reusing them instead of recomputing them several times.

In the end, for any bar b_k, the optimal segmentation until b_k can be computed in $\mathcal{O}(b_k - 1)$ operations, *i.e.* parsing each antecedent only once. Hence, the solution algorithm boils down to $\mathcal{O}(\frac{B(B+1)}{2})$ evaluations, which corresponds to a complexity in the order of $\mathcal{O}(B^2)$, and is polynomial. In practice, we even limit the size of admissible segments to be at most 32 bars, which further reduces the complexity.

Finally, the segmentation problem boils down to the definition of the score function $\gamma([\![\zeta_i, \zeta_{i+1}[\![)$ for a segment. In the CBM algorithm and following [SBV16], the score of each segment is defined as a mixed score function, presented in Equation 3.6 as the weighted sum of two terms: the first one, $\gamma^K([\![\zeta_i, \zeta_{i+1}[\![)$, is based on the homogeneity criterion, and is presented in Section 3.3.2; the second one, $p(\zeta_{i+1} - \zeta_i)$, is based on the regularity criterion, and is presented in Section 3.3.3. Parameter λ is a weighting parameter.

$$\gamma([\![\zeta_i, \zeta_{i+1}[\![) = \gamma^K([\![\zeta_i, \zeta_{i+1}[\![) - \lambda p(\zeta_{i+1} - \zeta_i). \tag{3.6}$$

3.3.2 Convolution Kernels

The first term $\gamma^K()$ of the score function defined in Equation 3.6 is obtained from the autosimilarity values in this segment. Practically, given an autosimilarity matrix $A(X)$, the score $\gamma^K(S_i)$ of the segment $S_i = [\![\zeta_i, \zeta_{i+1}[\![$ (of size $n = \zeta_{i+1} - \zeta_i$) is computed by evaluating the autosimilarity values restricted to the segment, *i.e.* $A(X_{:S_i}) = A(X_{[:,\zeta_i:\zeta_{i+1}-1]})$. It can be seen as cropping the autosimilarity $A(X)$ on this particular segment, around the diagonal, as shown in Figure 3.3.

The CBM algorithm aims at favoring the homogeneity of estimated segments, *i.e.* favoring sections composed of similar components. Thus, the score function $\gamma^K()$ is developed so as to represent, in some sense, the inner similarity of this segment. In practice, this is obtained by weighting the different values in the autosimilarity, in a convolution [10]

10. Even if the operation is not formally a convolution but rather a cross-correlation, we abusively define the operation as a convolution, representing the essence of the elementwise product of two matrices. In addition, as the autosimilarity matrix is symmetric, both the 2D convolution and 2D cross-correlation are equivalent.

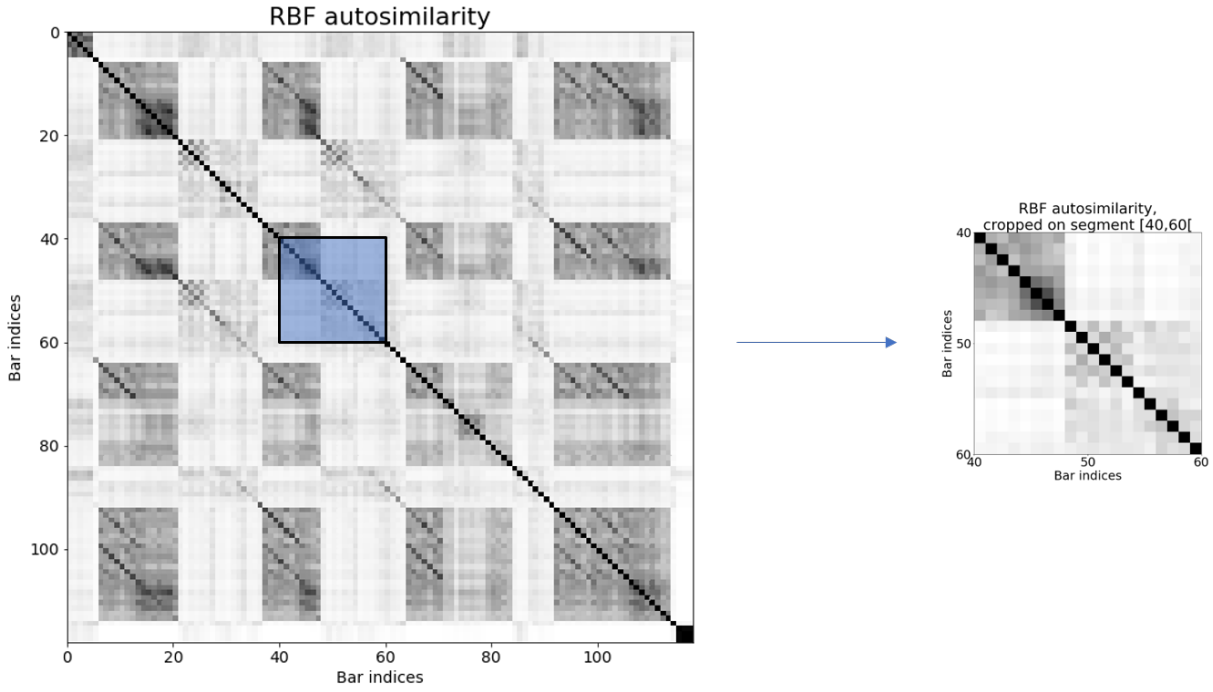


Figure 3.3 – Autosimilarity restricted to the segment $[\![40, 60[\![$.

operation between the autosimilarity and a (fixed) kernel matrix K, such as:

$$\begin{array}{rccc} \gamma^K & : & \mathbb{R}^{n\times n} & \to & \mathbb{R} \\ & & A(X_{:S_i}) & \mapsto & \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1}^{n} A(X_{:S_i})_{kl}K_{kl}. \end{array} \tag{3.7}$$

The kernel is called a "convolution kernel". A first observation is that the convolution kernel is exactly of the size of the autosimilarity, *i.e.* it adapts to the size of the segment.

A very simple kernel is a kernel matrix full of ones, *i.e.* $K = \mathbb{1}_{n\times n}$, resulting in a score function equal to the sum of every element in the autosimilarity, normalized by the size of the segment. The normalization by the size of the segment is meant to transform the squared dependence of the size of the segment in the number of autosimilarity values (n^2 values in the autosimilarity) in a linear dependence [11].

In a very similar work, Jensen [Jen06] defines the homogeneity value for a segment as an aggregated value of the similarities in this segment [12]. In details, the score for the

11. It can equivalently be seen as computing the average value, hence dividing the result of the convolution by n^2, and then multiplying this average value by the size of the segment n, *i.e.* weighting every bar constituting this segment with this average value.

12. In details, in their work, the autosimilarity values are defined in the sense of a "distance": the lower are the values in the autosimilarity, the more similar are the elements. Still, minimizing the distance

segment S_i in [Jen06] is computed as $\gamma(S_i) = \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1}^{k} A(X_{:S_i})_{kl}$. This score corresponds to the sum of the autosimilarity values in the lower triangle of the autosimilarity restricted to the segment (including the main diagonal), divided by the size of the segment. To some extent, it represents a score close to the average similarity value in the segment. It could be implemented in the CBM formulation by defining a convolution kernel $K_{kl} = \begin{cases} 1 & \text{if } k \geq l \\ 0 & \text{if } k < l \end{cases}$.

In the CBM algorithm, the design of the convolution kernel defines how to transform bar similarities into segment homogeneity, which is of particular importance. As opposed to the work of Jensen, we consider that the main diagonal in the autosimilarity is not informative regarding the overall similarity in the segment, as its values are normalized to one. In addition, it could even favor the shortest segments: any segment being composed of n^2 values, and the diagonal representing n values, the proportion of diagonal values in the segment is equal to $\frac{1}{n}$, which decreases with the size of the segment.

> The CBM algorithm does not take into account the main diagonal in the score computation, because it represents the self-similarities of each bar, which is not informative. Hence, for every convolution kernel K used in the CBM algorithm, $K_{ii} = 0$, $\forall i$.

The remainder of this section presents two types of kernels, namely the "full" kernel and the "band" kernel.

Full Kernel

The first kernel is called the **"full"** kernel, because it corresponds to a kernel full of 1 (except on the diagonal where it is equal to 0). The full kernel captures the average value of similarities in this segment, without the self-similarity values. Practically, denoting as K^f the full kernel:

$$K^f_{ij} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{3.8}$$

Hence, the convolution score function associated with the full kernel is equal to:

$$\gamma^{K^f}(S_i) = \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1}^{n} A(X_{:S_i})_{kl} K^f_{kl} = \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1,l\neq k}^{n} A(X_{:S_i})_{kl}. \tag{3.9}$$

is conceptually similar to maximizing the similarity, and, except for correctness, we do no pay closer attention to the differences in formulations.

A full kernel of size 10 is presented in Figure 3.4.

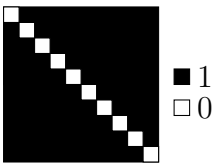


Figure 3.4 – Full kernel of size 10

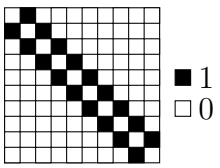
Band Kernels

A second class of kernels, called **"band"** kernels, are developed in order to emphasize on short-term similarity. Indeed, in band kernels, the convolution score is computed on a few bars in the segment only, depending on their temporal proximity: only close bars are considered. In practice, this can be obtained by defining a kernel with entries equal to 0, except on some upper- and sub-diagonals. The number of upper- and sub-diagonals is a parameter, corresponding to the maximal number of bars considered to evaluate the similarity, *i.e.* an upper bound on $|b_i - b_j|$ for a pair of bars (b_i, b_j).

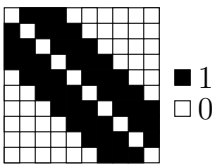
Hence, a band kernel is defined according to its number of bands, denoted as v, defining the v-bands kernel K^{vb} such that:

$$K^{vb}_{ij} = \begin{cases} 1 & \text{if } 1 \leq |i-j| \leq v, \\ 0 & \text{otherwise } (i = j \text{ or } |i-j| > v). \end{cases} \tag{3.10}$$

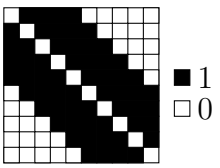
Three band kernels, of size 10, are represented in Figure 3.5.



(a) 1-band kernel K^{1b}.



(b) 3-bands kernel K^{3b}.



(c) 4-bands kernel K^{4b}.

Figure 3.5 – Band kernel, of size 10

3.3.3 Penalty Functions

Sargent *et al.* [SBV16] extended the score function of Jensen [Jen06] to take into account both the homogeneity and the regularity criteria, resulting in Equation 3.6. In practice, this is obtained by defining a penalty function $p(n)$, which corresponds to the second term in Equation 3.6, and is related to the regularity of the segment. More precisely, the regularity term penalizes segments according to their size n, in order to favor particular sizes.

In order to mitigate both the convolution score function $\gamma^K()$ and the penalty function $p()$, we implemented an additional normalization step based on the convolution values in this particular song. In details, this results in the score function:

$$\gamma([\![\zeta_i, \zeta_{i+1}[\![) = \gamma^K([\![\zeta_i, \zeta_{i+1}[\![) - \gamma^{K8}_{\max} \lambda p(\zeta_{i+1} - \zeta_i), \tag{3.11}$$

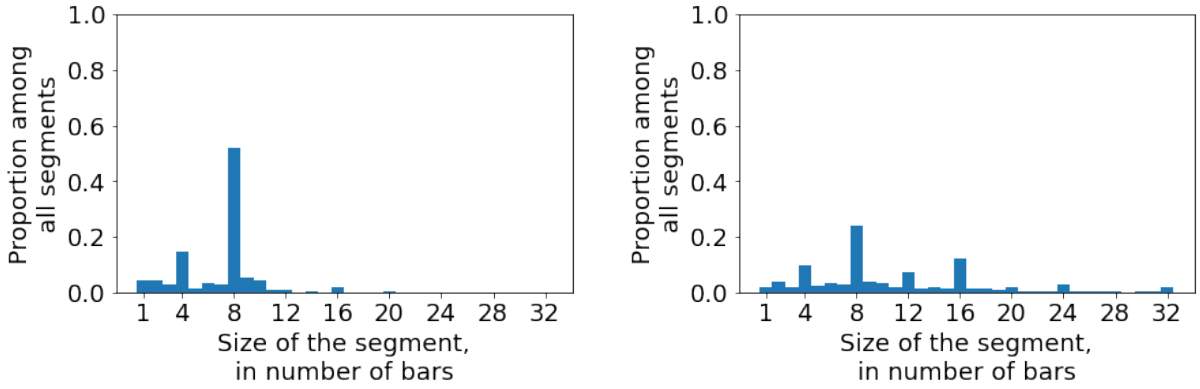
where $\gamma^{K8}_{\max}$ is the maximal convolution value obtained by sliding a kernel of size 8 on this autosimilarity, *i.e.* the highest score among all possible segments of size 8. This size of 8 for the kernel is chosen as the most common segment size in terms of number of bars in both RWC Pop and SALAMI datasets.

The penalty function is based on prior knowledge, and aims at enforcing particular sizes of segments, which are known to be common in a number of music genres, notably Pop music. In particular, some sizes of segments were shown to be most common in the annotations in Section 2.4.3. For convenience, the distributions of the sizes of segments, in terms of number of bars, in the annotations of both RWC Pop and SALAMI datasets, are presented again in Figure 3.6. Hence, penalty functions $p()$ can be inspired from these distributions.

Two different penalty functions $p()$ are studied in this section, namely the "target-deviation" and "modulo" functions. In what follows, n denotes the size of the segment, *i.e.* $n = \zeta_{i+1} - \zeta_i$.

Target-Deviation Functions

The first set of penalty functions, called "target-deviation" and denoted as $p^{td}()$, is defined by Sargent *et al.* [SBV16]. Target-deviation functions compute the difference between the size of the current estimated segment and a target size τ, raised to the power of a parameter α, *i.e.* $p^{td}(n) = |n - \tau|^\alpha$ where parameter α takes typical values in $\{0.5, 1, 2\}$. In the work of Sargent *et al.* the target size is set to 32, to favor segments of size 32 beats,



(a) RWC Pop.

(b) SALAMI (test subset).

Figure 3.6 – Distribution of segments sizes in terms of number of bars, in the annotations.

in line with their evaluations of most common segment sizes. In our barwise context [13], $\tau = 8$, which is the most common segment size in both RWC Pop and SALAMI datasets.

This penalty function is adapted to enforce one size in particular, and tends to disadvantage all the others. Hence, this function is adapted to datasets where one size is predominant among the reference, which seems true for RWC Pop with MIREX 10 annotations (where more than half of the segments in the annotation are of size 8), but not so definite for the SALAMI dataset, where the segment sizes are more balanced between 4, 8, 12 and 16, as presented in Figure 3.6. In particular, segments of size 16 are strongly penalized.

Modulo function

The second set of penalty functions, called "modulo functions", is designed to favor particular segment sizes, directly based on prior knowledge. In this study, we only present the "modulo 8" function $p^{m8}(n)$ based on both RWC Pop and SALAMI annotations. Indeed, in both dataset, most segments are of size 8, and the remaining segments are generally of size 4, 12 or 16. Finally, outside of these sizes, even segments are more common

13. Note that 8 bars containing 4 beats each leads to 32 beats. 4 beats per bar is a common value for Pop music.

than segments of odd sizes. Hence, the modulo 8 function models this distribution, as:

$$p^{m8}(n) = \begin{cases} 0 & \text{if } n = 8 \\ \frac{1}{4} & \text{else, if } n \equiv 0 \pmod 4 \\ \frac{1}{2} & \text{else, if if } n \equiv 0 \pmod 2 \\ 1 & \text{otherwise} \end{cases} \tag{3.12}$$

Penalty values for the different cases were set quite empirically, and would benefit from further investigations.

3.4 Experiments

The CBM algorithm being a segmentation algorithm, it is evaluated on the structural segmentation task on both RWC Pop and SALAMI datasets. In details, following the guideline defined in Appendix A.2, the experiments are conducted on the test set of the SALAMI dataset.

The CBM algorithm is applied on autosimilarity matrices, which are computed from the Barwise TF matrix. In this set of experiments, we focus on Barwise TF matrices computed in the Log Mel feature, as it is the feature used in the current State-of-the-Art algorithm [GS15b].

Several similarity functions for the computation of autosimilarities are introduced, namely the Cosine, Covariance and RBF autosimilarity matrices. The CBM algorithm itself is subject to the choice of the kernel, and particularly to the number of bands when using a band kernel. In addition, the score function depends on the design of the penalty function.

Rather than studying all of these parameters at the same time, experiments focus on each aspect independently. In particular, the experiments aim at answering the three following questions:

Question 1 *Which similarity function is the most adapted to the segmentation of the Barwise TF autosimilarity matrices?*

Question 2 *Which convolution kernel is the most adapted to the segmentation of the Barwise TF autosimilarity matrices?*

Question 3 *Which penalty function is the most adapted to the segmentation of the Barwise TF autosimilarity matrices?*

Each question is evaluated sequentially, and the conclusion of each question serves as the basis to study the following ones.

3.4.1 Autosimilarity Matrices

Firstly, we study the impact of the design of the similarity function (and, hence, of the autosimilarity) on the performance of the CBM algorithm. To do so, we use the CBM algorithm with the full kernel, as it does not need the fitting of the number of bands. In addition, we do not use penalty function.

Segmentation results are presented in Tables 3.1 and 3.2 respectively for the RWC Pop and SALAMI datasets, and according to the metrics defined in Section 2.3.4, *i.e.* $P_{0.5}$, $R_{0.5}$, $F_{0.5}$ and P_3, R_3, F_3. For both datasets, the RBF autosimilarity is the best-performing autosimilarity in terms of F measure (in both tolerances), hence suggesting a better segmentation in average than the other similarity functions.

Autosimilarity	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Cosine	**62.40%**	33.46%	43.17%	**81.83%**	43.95%	56.66%
Covariance	51.44%	**62.00%**	55.35%	64.61%	**77.89%**	69.54%
RBF	60.31%	54.02%	**56.25%**	77.27%	68.84%	**71.86%**

Table 3.1 – Results when computing different autosimilarities on the Barwise TF matrix, on the RWC Pop dataset. (Full kernel, no penalty function.)

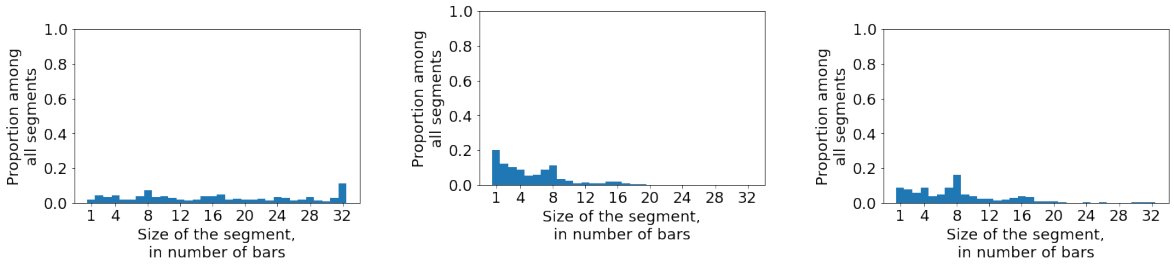
Autosimilarity	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Cosine	**47.61%**	34.11%	38.74%	**64.86%**	46.44%	52.76%
Covariance	27.98%	**58.39%**	36.37%	40.29%	**85.13%**	52.58%
RBF	44.22%	45.99%	**43.75%**	62.84%	65.88%	**62.41%**

Table 3.2 – Results when computing different autosimilarities on the Barwise TF matrix, on the SALAMI dataset. (Full kernel, no penalty function.)

The precision/recall trade-offs depend on the autosimilarity matrices, and deserves to be studied to give further information on the quality of the segmentations. The Cosine autosimilarity results in a higher precision than recall on average, which suggests an under-segmentation, *i.e.* estimating too few boundaries (a large part of estimated boundaries

are correct, but a large part of boundaries from the annotations are not estimated). Conversely, the Covariance autosimilarity results in a higher recall than precision, suggesting over-segmentation. The RBF autosimilarity performance is more balanced between both metrics.

These conclusions can be confirmed by studying the distribution of the sizes of the estimated segments, as presented in Figure 3.7 for the RWC Pop dataset. These distributions must be compared with the distribution of segment sizes in the annotation, presented in Figure 3.6.



(a) Cosine autosimilarity.

(b) Covariance autosimilarity.

(c) RBF autosimilarity.

Figure 3.7 – Distribution of the segment sizes, with the full kernel, according to the autosimilarity matrix. Results are computed on the RWC Pop dataset.

While the segments in the annotation are predominantly of size 8 (and, secondarily, 4), the most common estimated sizes for the Cosine and Covariance autosimilarity matrices are respectively 32 bars [14] and 1 bar. A perfect estimation would compute a distribution close to those of the annotations, while no segment distribution for estimated segmentations is indeed following the annotation's distribution here. The distribution of segment sizes with the RBF autosimilarity is visually the closest one to the distribution in annotation, again suggesting that this similarity function is the most adapted.

We explain these differences in segmentation by contrast differences in the different autosimilarity matrices. The contrast in an autosimilarity matrix represents the value differences between zones of high similarity and low similarity and, empirically, autosimilarity matrices represented in Figure 3.1 exhibit differences in contrast.

In our context, the CBM algorithm is designed so as to frame homogenous regions (*i.e.* blocks). In that spirit, the more contrastive are regions of high and low similarity, the less

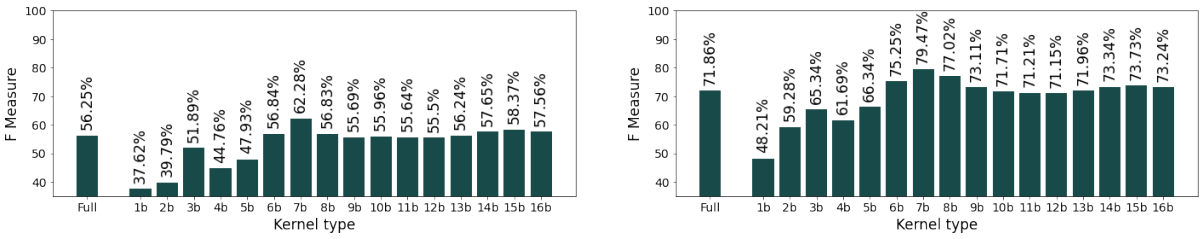
14. Note that 32 bars is the largest admissible size of segments in our algorithm. As a test, we extended the largest admissible segment size to 36 bars and observed that the most common segment size was 36. Hence, it seems that the full kernel favors the longest admissible segments.

ambiguous is the block structure in the matrix and the larger are the differences between the convolution values on zones of high and low similarity. Hence, increasing the contrast in an autosimilarity matrix should improve the retrieval of the optimal segmentation in the CBM algorithm.

In what follows, we hence study only the RBF autosimilarity function.

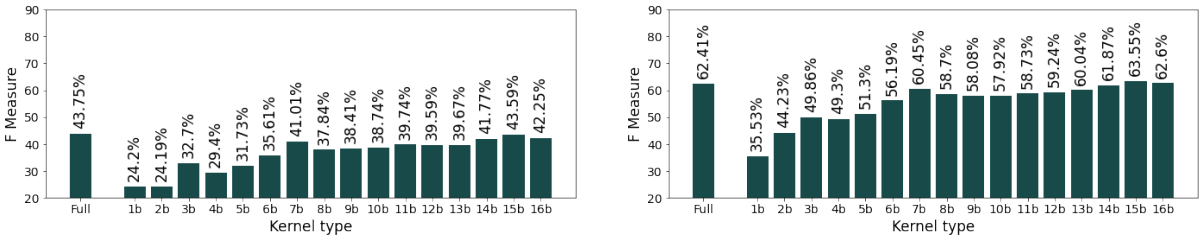
3.4.2 Convolution Kernels

Secondly, an important parameter in the CBM algorithm is the design of the kernel. We thus compare the full kernel with band kernels, the number of bands varying from 1 to 16 bands. Results, focusing on the F-measures, are presented in Figures 3.8 and 3.9 respectively for the RWC Pop and SALAMI datasets. The 7-bands kernel stands out for the RWC Pop as the best-performing kernel, while the 15-bands is the best one for SALAMI.



(a) $F_{0.5}$. (b) F_3.

Figure 3.8 – Comparison of the F measures, according to the full and band kernels (with different number of bands). Results are computed on the RWC Pop dataset.



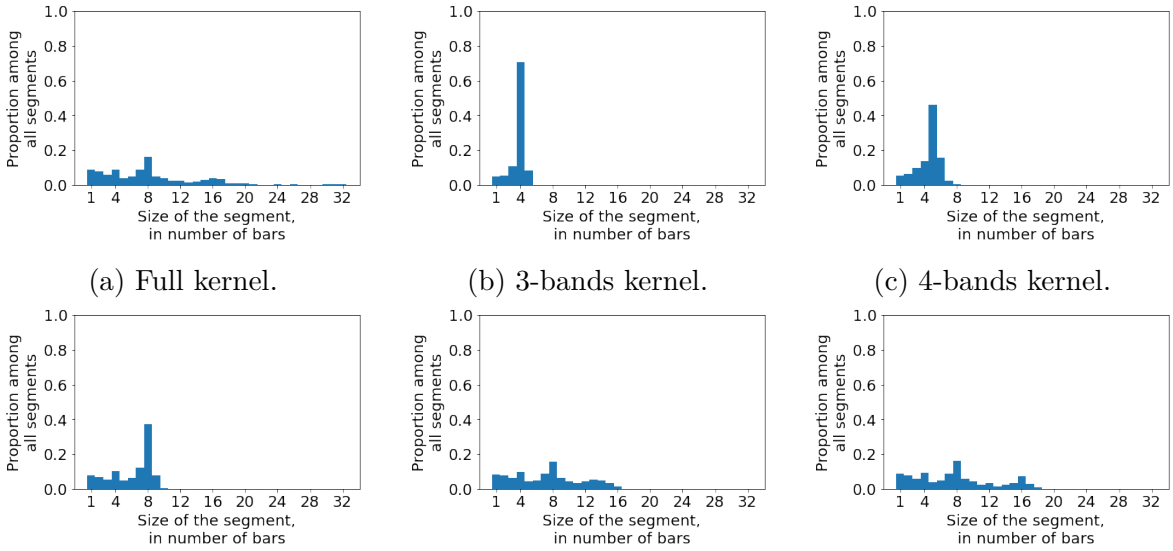
(a) $F_{0.5}$. (b) F_3.

Figure 3.9 – Comparison of the F measures, according to the full and band kernels (with different number of bands). Results are computed on the SALAMI dataset.

Improved performance of the kernels may be explained by looking at the distribution of segment sizes in the estimation, in Figures 3.10 and 3.11 respectively for the RWC

Pop and SALAMI datasets. In both datasets, the 7-bands kernel leads to a majority of estimated segments of size 8 (more than 50% for SALAMI). For the RWC Pop dataset, this is a major asset, as this is also the most common size in the annotation.

Conversely, for the SALAMI dataset, the 7-bands kernels is not able to estimate accurately segments larger than 8 bars (for 12 and 16 bars in particular, which are present in the annotation), and estimates twice as much segments of size 8 as in the annotation, in proportion. As for the 15-bands kernel, estimated segments are mostly of size 16 bars. Hence, while it does not accurately represent the annotation, better segmentation results are obtained with the 15-bands kernels, indicating that this latter distribution is beneficial to the segmentation overall. The full kernel obtains similar performance than the 15-bands kernel, while it was performing less well on RWC Pop.

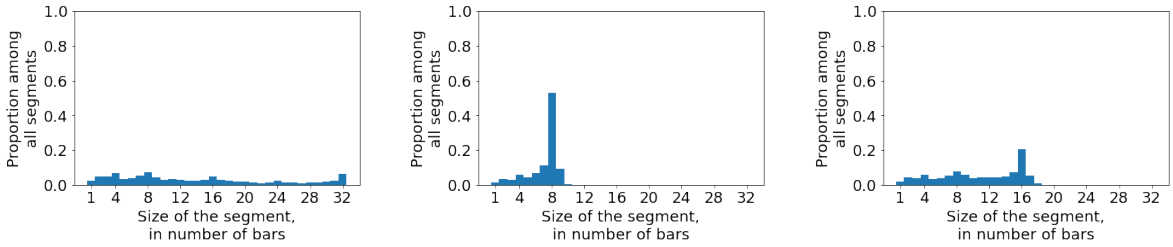


(a) Full kernel. (b) 3-bands kernel. (c) 4-bands kernel.

(d) 7-bands kernel. (e) 12-bands kernel. (f) 15-bands kernel.

Figure 3.10 – Distribution of the segment sizes, with the RBF autosimilarity matrix, according to different kernels. Results are computed on the RWC Pop dataset.

As an additional conclusion, the number of bands in the kernel largely influences the distribution of segment sizes in the annotation, in particular the most common segment size. As a general trend, it seems that a kernel with v bands favors segments of size $v + 1$. We assume that this behavior stems from the fact that, for a v-bands kernel and a large segment of size $n > v$, the number of elements equal to 0 is important, but the normalization remains adapted to kernels with n^2 values. Adapting the normalization in the kernel to the number of nonzero values, as in the work of Shiu *et al.* [SJK06], could



(a) Full kernel. (b) 7-bands kernel. (c) 15-bands kernel.

Figure 3.11 – Distribution of the segment sizes, with the RBF autosimilarity matrix, according to different kernels. Results are computed on the SALAMI dataset.

possibly reduce the effect of the number of bands in the sizes of estimated segments. This lead was not explored to this day, and may constitute future work.

Finally, for the rest of this study, we conclude that the 7-bands kernel is the optimal kernel for the RWC Pop dataset, and that the 15-bands kernel is the most adapted for the SALAMI dataset. We thus fixed these kernels for these datasets.

3.4.3 Penalty Functions

Finally, the last experiments focus on the penalty functions. In this set of experiments, we compare the target deviation functions, with $\alpha \in \{0.5, 1, 2\}$, with the modulo 8 function. The CBM algorithm is parametrized with the 7-bands and 15-bands kernels respectively for the RWC Pop and SALAMI datasets, and is applied on the RBF autosimilarity matrices. The parameter λ, weighting the penalty function, takes values between $\frac{1}{10}$ and 2, with a step of $\frac{1}{10}$. This parameter is fitted by cross-validation on the RWC Pop and on the learning dataset for SALAMI, as precisely defined in Appendix A.2.

Results are presented in Tables 3.3 and 3.4 respectively for the RWC Pop and SALAMI datasets. In both conditions, the modulo 8 function appears to improve segmentation results, in particular for the metrics with 0.5s tolerance. A good estimation with a tolerance of 0.5s indicates that the estimated boundary is precisely located on the same bar than an annotated one. Hence, a better performance with a 0.5s tolerance indicates a more accurate estimation. Results with the 3s tolerance are not strongly impacted by the choice of the penalty function, except for the target deviation with a large α, obtaining worst performance than the other conditions.

Overall, it seems that the modulo 8 function is the most adapted penalty function to estimate segments more accurately.

Penalty function		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Without penalty		59.09%	67.13%	62.28%	75.17%	**85.91%**	79.47%
Target deviation	$\alpha = \frac{1}{2}$	63.43%	64.90%	63.70%	80.22%	82.55%	80.80%
	$\alpha = 1$	62.70%	62.06%	61.94%	**81.51%**	81.20%	80.79%
	$\alpha = 2$	56.48%	53.03%	54.30%	78.09%	73.63%	75.27%
Modulo 8		**64.32%**	**70.01%**	**66.52%**	78.31%	85.64%	**81.16%**

Table 3.3 – Segmentation results depending on the penalty function, for the RWC Pop dataset, with the RBF autosimilarity and the 7-bands kernel.

Penalty function		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Without penalty		40.84%	49.45%	43.59%	59.43%	72.24%	63.55%
Target deviation	$\alpha = \frac{1}{2}$	41.35%	49.96%	43.96%	**60.07%**	72.69%	**63.93%**
	$\alpha = 1$	38.85%	50.10%	42.55%	56.92%	**74.00%**	62.54%
	$\alpha = 2$	33.40%	47.08%	37.84%	51.11%	72.18%	57.96%
Modulo 8		**42.56%**	**50.30%**	**44.94%**	59.66%	70.99%	63.18%

Table 3.4 – Segmentation results depending on the penalty function, for the SALAMI dataset, with the RBF autosimilarity and the 15-bands kernel.

3.4.4 Experimental Conclusions

In light of these results, we finally conclude regarding the choice of settings in the CBM algorithm.

Experimental conclusion 1 *On studied datasets, the RBF autosimilarity matrix is the most adapted autosimilarity matrix for the segmentation of the Barwise TF autosimilarity matrices.*

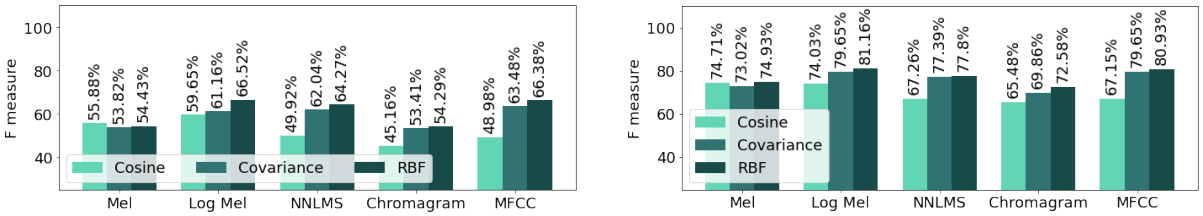
Experimental conclusion 2 *Both 7-bands and 15-bands kernels are the most adapted kernels for the segmentation the Barwise TF autosimilarity matrices, respectively for the RWC Pop and SALAMI datasets.*

Experimental conclusion 3 *On studied datasets, the modulo 8 penalty function is the most adapted penalty function for the segmentation of the Barwise TF autosimilarity matrices.*

From these experimental conclusions, we decided to fix the CBM algorithm in terms of convolution kernel and penalty function to the best of conditions obtained here. Indeed, subsequent chapters will use the CBM algorithm, and thus reuse these settings. The impact of the different autosimilarity matrices will be studied again.

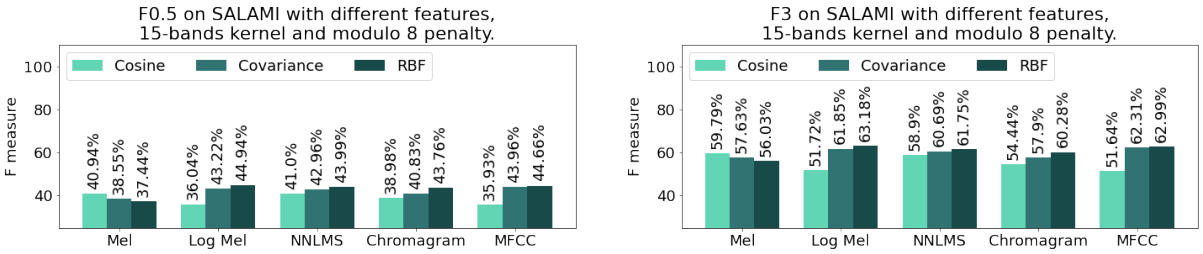
Figures 3.12 and 3.13 present the results of the CBM algorithm depending on the different features, and depending on the different autosimilarity matrices. Results confirm that the choice of both autosimilarity matrices and feature representation impact the performance of segmentation.

Figures 3.14 and 3.15 compare the best results obtained with the CBM algorithm with those of the State-of-the-Art (aligned on downbeats, as presented in Figure 2.14). In this comparison, the CBM algorithm largely outperforms the other blind segmentation methods, and is competitive with the global (supervised) State-of-the-Art [GS15b].



(a) $F_{0.5}$. (b) F_3.

Figure 3.12 – Segmentation results depending on the choice of the feature description and on the autosimilarity matrix. Results are computed on the RWC Pop dataset, with the 7-bands kernel and the modulo 8 penalty function.



(a) $F_{0.5}$. (b) F_3.

Figure 3.13 – Segmentation results depending on the choice of the feature description and on the autosimilarity matrix. Results are computed on the SALAMI dataset, with the 15-bands kernel and the modulo 8 penalty function.

3.5 Conclusions

This chapter has presented different autosimilarity matrices, studying several distinctive ways to represent similarities between pairs of bars in a song. These autosimilarities

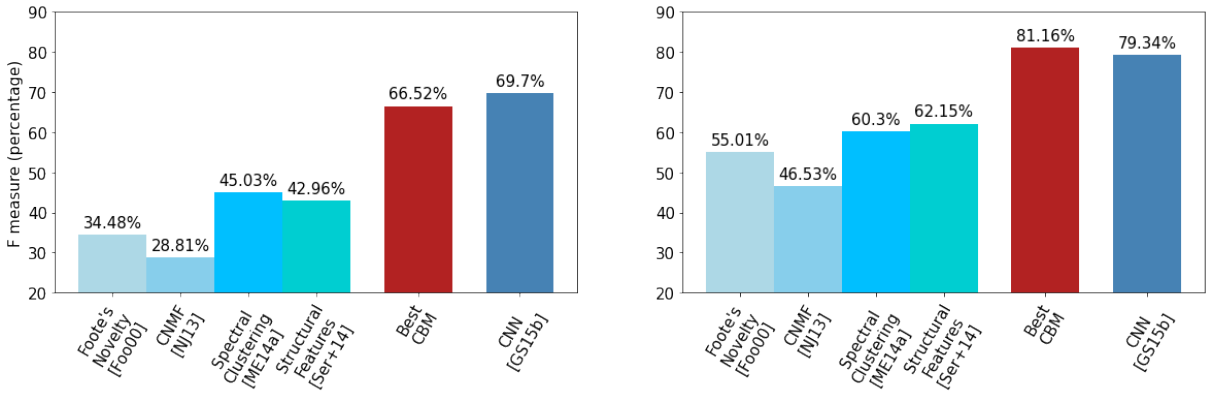


Figure 3.14 – Segmentation results of the best CBM algorithm, compared to the other State-of-the-Art algorithms. Results are computed on the RWC Pop dataset.

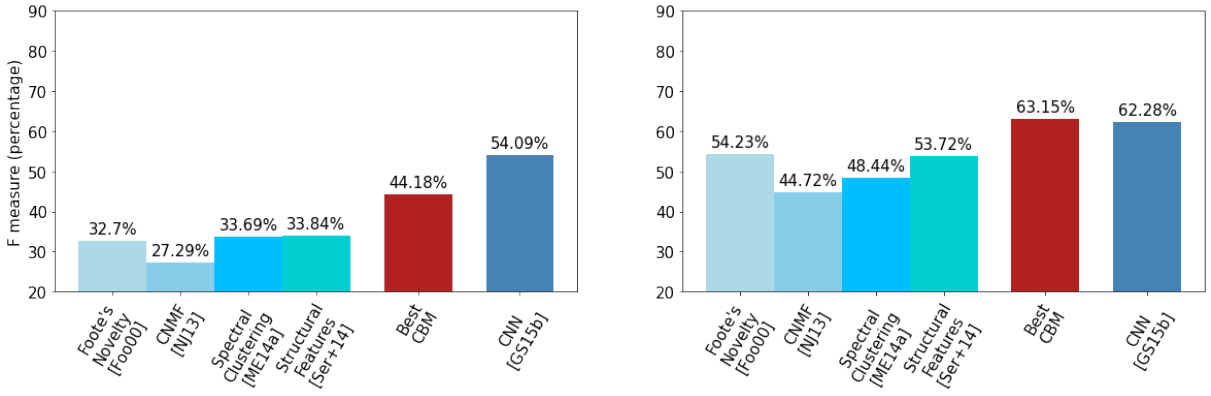


Figure 3.15 – Segmentation results of the best CBM algorithm, compared to the other State-of-the-Art algorithms. Results are computed on the SALAMI dataset.

are at the heart of the segmentation strategy, through the CBM algorithm. Boundaries between sections are computed by maximizing the homogeneity (in some sense) of each segment composing the segmentation, using dynamic programming.

The CBM algorithm is introduced here with several settings, which have been partly discussed. This algorithm could still be improved, but it already achieves levels of performance comparable to those of the State-of-the-Art [GS15b], which makes of it a useful tool to investigate a variety of music representations. We strongly believe that the barwise process of music is one of the main reasons for explaining these performance.

Overall, the design of the kernel clearly impacts the segmentation results. Hence, future work could focus on studying alternative types of kernels. The kernel values could depend on the particular song or dataset considered, or, following the developments of the kernel

of Foote [Foo00], values could be taken following a Gaussian distribution. Of particular interest could be the learning of such kernels instead of an (empirical) definition.

Convolution kernels presented in this section focus on the homogeneity of each segment, but different kernels, in particular considering those of Shiu *et al.* [SJK06], could be considered in order to account for repetition in the song.

Finally, the number of bands in the convolution kernels seems to enforce some segment sizes in particular. This could be further exploited, for instance by using different kernels concurrently, each one accounting for a different level of structure, hence studying segmentation hierarchically.

A major problem observed with the current version of the CBM algorithm is some lack of robustness: we observed in practice that very similar autosimilarity matrices could result in different segmentations, sometimes inconsistent with one another. One way which was investigated to counteract this effect consisted in adding a second penalization in the cost, proportional to the largest convolution score in the current row. The rationale was to favor segments when a block of high similarity appeared in the same row, hence indicating a repetition of this segment in previous or subsequent bars.

Unfortunately, this idea was not sufficiently beneficial segmentation-wise compared to the loss in complexity (numerous convolutions across the whole row instead of only convolution around the diagonal). Still, this kind of score could be highly informative to stabilize the outputs, and/or consider repetitions, or in a labelling phase for instance.

Additionally, the lack of robustness could be related to the observed impact of the number of bands in the distribution of the estimated segment sizes. We conjecture that counteracting this effect, for instance by using normalized values in the kernel (as in [SJK06]) and normalizing the score associated with each kernel by the number of nonzero values instead of the size of the kernel could make the algorithm more robust.

Subsequent chapters will study how **compressed representations** of bars can provide relevant features for structural segmentation. Indeed, instead of computing the autosimilarity matrices on the barwise feature representation of the song, they can be computed on barwise compressed representations and be used as the input of the CBM algorithm. These compressed representations will largely impact the similarity values between the different bars of the song, and notably the contrasts between zones of high and low similarity.

Part II

Barwise Compression Methods

Chapter 4

Nonnegative Tucker Decomposition - NTD

Synopsis

This chapter presents the Nonnegative Tucker Decomposition, a tensor factorization scheme, algorithms to compute it, and presents its use for structural segmentation and the discovery of patterns in music.

4.1 Introduction

Nonnegative Tucker Decomposition (NTD) is a nonnegative tensor factorization technique introduced by Kim and Choi [KC07]. NTD computes a part-based representation of the (nonnegative) tensor of data, and generally serves as a multilinear dimensionality reduction technique. NTD is a powerful tool to extract relevant patterns in data in an unsupervised fashion, and is nowadays used in numerous applications, see for instance [ZF22] for a recent overview of applications. NTD is computed by solving an optimization problem, subject to a loss function.

NTD is used in this thesis as a barwise compression method, which can help to reveal structure in music. In addition, NTD is able to extract interpretable musical patterns at the barscale, showcased on a task of pattern uncovering.

This chapter first introduces the mathematical aspects of NTD, along with some primordial notions of tensor algebra. In a second part, algorithms to compute NTD are introduced, relatively to different loss functions: the Euclidean distance and the family of β-divergences. Finally, NTD is presented in the context of Music Information Retrieval, from a theoretical point of view firstly, and in an experimental phase secondly. In particular, experiments are made on the structural segmentation task, defined in previous chapters, and in a new task of pattern uncovering, showcasing numerous potential applications for NTD.

The contributions reported in this section are threefold:

— Algorithmic: Two NTD algorithms are presented (one optimizing the Euclidean distance with HALS, one optimizing the β-divergence with MU) in the context of this thesis. Both algorithms were developed by the authors, based on existing algorithms used for other low-rank approximations (mainly Nonnegative Matrix Factorizations).

— Experimental: NTD was previously used in MIR in a pattern uncovering task [SG18], and we extend its use to structural segmentation. Experimental performance of NTD outperform the unsupervised State-of-the-Art techniques, and obtain, to some extent, comparable results with those of the current supervised State-of-the-Art [GS15b].

— Pedagogical: an important part of this work lies in pedagogical efforts to interpret NTD, which may be hard to decipher when introduced to a novel reader.

These contributions have also resulted in two publications [Mar+20; Mar+22]. The

NTD algorithms are included in the open-source *nn_fac* toolbox [MC20], and the remainder of the code, along with the conducted experiments, are included in the open-source *BarMusComp* toolbox [MCB22b].

4.2 Elements of Tensor Algebra

Let us start by introducing tensors and essential notions of tensor algebra. This section does not cover exhaustively tensor algebra, only primordial concepts to rigorously introduce NTD.

4.2.1 Tensors

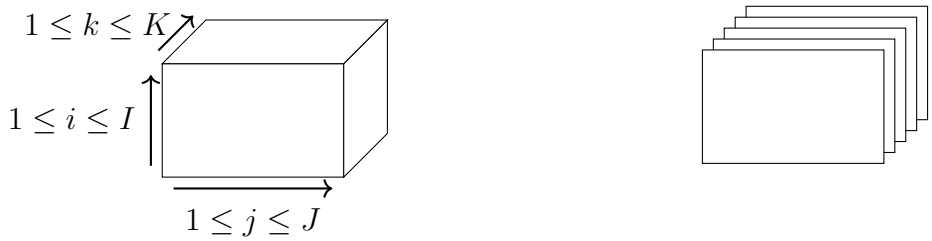
Generally speaking, a tensor is a multi-dimensional array. Citing [KB09], "formally, a N-way or N^{th}-order tensor is an element of the tensor product of N vector spaces". It means that tensors are the generalization of both vectors (1^{st}-order tensor) and matrices (2^{nd}-order tensor). In this thesis, tensors are studied up to the 3^{rd}-order, even if the algorithms and algebra presented can be applied to arbitrary N^{th}-order tensors. Hence, we further simplify the notations by denoting 3^{rd}-order tensors as "tensors".

Notation-wise, tensors are represented by Euler script letters, such as $\mathcal{X}$, and any element (i, j, k) of a tensor is denoted as $\mathcal{X}_{ijk}$. In this thesis, tensors are restricted to real-valued tensors, generally nonnegative, thus $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$or $\mathbb{R}_{+}^{I \times J \times K}$.

Each of the three dimensions of a tensor is called a **mode**. When fixing the index along one mode, *e.g.* $\mathcal{X}_{i::}$ on the first mode (a colon represents all the elements in a mode), the resulting subarray has two modes, and can be seen as a matrix, called **slice** of the tensor. When fixing two modes, such as $\mathcal{X}_{ij:}$, the resulting subarray is a vector, called **fiber** of the tensor. A tensor and its representation in slices are presented in Figure 4.1.

Matricization - Unfolding

A tensor can be transformed into a matrix via the **mode-n unfolding** operation. Practically, the idea is to keep one mode intact, and to reorder all the elements of the other modes into a unique mode, *i.e.* concatenate all the fibers of a tensor in a second mode. For instance, given the tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, it can be transformed into matrices $\mathcal{X}_{(1)} \in \mathbb{R}^{I \times JK}, \mathcal{X}_{(2)} \in \mathbb{R}^{J \times IK}$or $\mathcal{X}_{(3)} \in \mathbb{R}^{K \times IJ}$, respectively by unfolding on the first, second or third mode. As presented with the previous example, the mode-n unfolding of a tensor



(a) Tensor, with (i, j, k) indexing the three modes.

(b) Tensor, evaluated by its different slices on the last mode: $\mathcal{X}_{::k} \forall 1 \leq k \leq K$.

Figure 4.1 – A tensor, seen with its 3 modes and in slices.

$\mathcal{X}$ is denoted as $\mathcal{X}_{(n)}$. The inverse operation is called **folding**, *i.e.* obtaining a tensor from a matrix by reordering its elements.

The order for unfolding is not subject to a consensus in the community. In this thesis, we choose to reorder elements by taking them in the descending order of their index [Coh15]. Practically, the mode-1 unfolding of tensor $\mathcal{X}$ takes elements on the 3rd mode first and on the 2nd mode last.

Similarly, the **vectorization** operation $\text{vec}(\mathcal{X}) \in \mathbb{R}^{IJK}$ can be defined, in the same order (from the highest order to the smallest one). Illustrations are given in Figure 4.2.

(a) Schematic representation of the order for unfolding/vectorizing, taken from [Coh15].

(b) Example of vectorization with a $2 \times 2 \times 2$ tensor.

Figure 4.2 – Order for unfolding and vectorization.

Kronecker and n-mode products

The Kronecker product is a matrix product, between $A \in \mathbb{R}^{a_1 \times a_2}$ and $B \in \mathbb{R}^{b_1 \times b_2}$, denoted as $\otimes$. The Kronecker product multiplies each entry in A with all entries in B,

hence $A \otimes B \in \mathbb{R}^{a_1 b_1 \times a_2 b_2}$:

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1a_2}B \\ \vdots & \ddots & \vdots \\ a_{a_1 1}B & \cdots & a_{a_1 a_2}B \end{pmatrix}. \tag{4.1}$$

Equation 4.2 presents the Kronecker product elementwise for 2×2 and 2×3 matrices:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \left(\begin{array}{ccc|ccc} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} \end{array}\right). \tag{4.2}$$

The Kronecker product is distributive with respect to transposition:

$$\begin{aligned} (A \otimes B)^\intercal &= A^\intercal \otimes B^\intercal, \\ (A \otimes B)^\intercal (A \otimes B) &= A^\intercal A \otimes B^\intercal B. \end{aligned} \tag{4.3}$$

The (matrix) n-mode product, denoted as $\times_n$, corresponds to the product between a matrix and a tensor on mode n. Practically, given a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ and a matrix $A \in \mathbb{R}^{D \times I}, (\mathcal{X} \times_1 A) \in \mathbb{R}^{D \times J \times K}$. Elementwise, $(\mathcal{X} \times_1 A)_{d,j,k} = \sum_{i=1}^{I} \mathcal{X}_{i,j,k} A_{d,i}$. It can also be seen as a matrix product when unfolding tensors:

$$\mathcal{T} = \mathcal{X} \times_1 A \Leftrightarrow \mathcal{T}_{(1)} = A\mathcal{X}_{(1)}. \tag{4.4}$$

The Kronecker and n-mode products can be linked through the Equations 4.5 and 4.6 listed below.

$$\mathcal{T} = \mathcal{P} \times_1 A \times_2 B \times_3 C \Leftrightarrow \operatorname{vec}(\mathcal{T}) = (A \otimes B \otimes C) \operatorname{vec}(\mathcal{P}) \tag{4.5}$$

$$\mathcal{T} = \mathcal{P} \times_1 A \times_2 B \times_3 C \Leftrightarrow \mathcal{T}_{(1)} = A\mathcal{P}_{(1)}(B \otimes C)^\intercal \tag{4.6}$$

4.2.2 Nonnegative Tucker Decomposition Model

Nonnegative Tucker Decomposition (NTD) is a nonnegative tensor factorization technique [KC07] in the framework of which a nonnegative tensor is approximated as the product of factors (one for each mode of the tensor) and a small core tensor linking these factors. This decomposition results in a low-rank approximation of the original tensor, which can also be seen as a projection of the original tensor in the multilinear space spanned by the factors.

NTD is often used as a dimensionality reduction technique, but it may also be seen as a part-based representation (as we will present in Section 4.4) although its identifiability properties are still not fully understood [Zho+15]. Computing the NTD means seeking for three nonnegative matrices $W \in \mathbb{R}_+^{I \times I'}$, $H \in \mathbb{R}_+^{J \times J'}$ and $Q \in \mathbb{R}_+^{K \times K'}$ and a core tensor $\mathcal{G} \in \mathbb{R}_+^{I' \times J' \times K'}$ such that:

$$\mathcal{X} \approx \mathcal{G} \times_1 W \times_2 H \times_3 Q \tag{4.7}$$

which rewrites using element-wise notation:

$$\mathcal{X}(i,j,k) \approx \sum_{i',j',k'=1}^{I',J',K'} \mathcal{G}(i',j',k') W(i,i') H(j,j') Q(k,k'). \tag{4.8}$$

Figure 4.3 depicts a schematic 3-D representation of an NTD. NTD core dimensions I', J' and K' are assumed to be known (or set empirically) prior to the decomposition. Core dimensions are sometimes loosely referred to as "ranks" of the decomposition (as in our previous work [Mar+20]). Mathematically, the core dimensions are related to, but not a trivial extension of the multilinear ranks of the core tensor. Discussion about multilinear ranks and NTD is to be found in [Ale+22].

NTD is performed by minimizing a loss function, which is generally a distance or divergence function $d()$ between the original tensor and the approximation. It leads to the following optimization problem:

$$W^*, H^*, Q^*, \mathcal{G}^* = \underset{W \geq 0, H \geq 0, Q \geq 0, \mathcal{G} \geq 0}{\arg\min} d(\mathcal{X}, \mathcal{G} \times_1 W \times_2 H \times_3 Q), \tag{4.9}$$

i.e. computing factors such that the error between the original data and the approximation in factors (reconstruction error) is minimized [1].

Many algorithms are devoted to the optimization of the Euclidean distance between

1. Notation "arg min" means finding the factors minimizing the error.

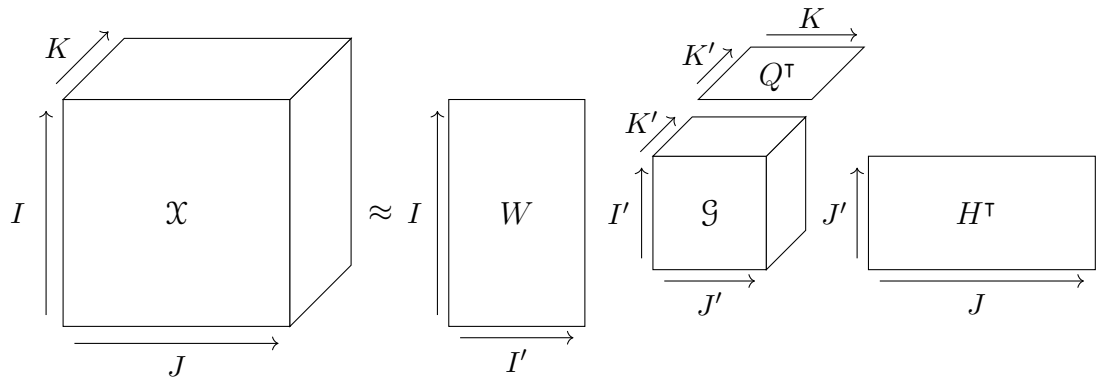


Figure 4.3 – Nonnegative Tucker Decomposition of tensor $\mathcal{X}$ in factor matrices W, H, Q, and core tensor $\mathcal{G}$, with their dimensions.

the original tensor and the factors [Zho+15; PC11; Xu15; Mar+20], as for the HALS algorithm, presented in Section 4.3.2. Other algorithms optimize the α- [KCC08] and β- [Mar+22] divergences. This thesis does not consider α-divergences, but β-divergences and an associated algorithm are presented in Section 4.3.3.

4.2.3 Nonnegative Matrix Factorization - NMF

Nonnegative Matrix Factorization (NMF) is a matrix decomposition model, resulting in part-based representation of the original data [LS99]. Starting from a nonnegative matrix $M \in \mathbb{R}_+^{n \times m}$, and a dimension hyperparameter r, NMF aims at finding two matrices $U \in \mathbb{R}_+^{n \times r}$ and $V \in \mathbb{R}_+^{r \times m}$ such that:

$$M \approx UV. \tag{4.10}$$

As for NTD, the problem is generally defined as an optimization problem:

$$U^*, V^* = \underset{U \geq 0, V \geq 0}{\arg\min}\, d(M, UV), \tag{4.11}$$

with $d()$ the loss function, *i.e.* some distance or divergence. Even if NTD and NMF are different problems, we will see in Section 4.3 that both can be solved using the same methods. In addition, NTD can be seen as a generalization of NMF.

4.3 Algorithms for NTD

This section presents the different algorithms developed during this thesis to compute NTD. They are based on the resolution of the NTD optimization problem subject to the Euclidean distance and β-divergences ($d()$ in Equation 4.9). Algorithms are iterative, hence iteration t for the update of a factor A is denoted as $A^{(t)}$.

Both of these algorithms are present in the open-source *nn_fac* toolbox [MC20], developed in the context of this thesis. Tensors (*i.e.* the data arrays) are computed and handled with the *Tensorly* toolbox [Kos+19]. Algorithms presented in this section should be deployed in the *Tensorly* toolbox in near future thanks to the work of my colleague C. Tuna.

4.3.1 Alternating Scheme

In general, algorithms used to solve NTD face the fact that the loss function defined in Equation 4.9 is non-convex with respect to all factors. In addition, finding a global solution to this equation is NP-Hard when the loss function is the Euclidean distance or a β-divergence [2]. These conclusions stem from results obtained for NMF [Vav10], which is a particular case of NTD.

> Therefore, NTD (and, analogously, NMF), is generally solved using alternating strategies: solving the problem with respect to one factor at a time (for instance, W) by fixing all the other factors (here, H, Q and $\mathcal{G}$).

Each alternating subproblem is easier to solve, and is even convex for the Euclidean distance and some of the β-divergences (when $\beta \in [1, 2]$). Hence, a general routine for solving both NMF and NTD is to alternate between factors, solving each subproblem independently.

When fixing one of the matrix factors (W, H or Q), NTD can be recast into a matrix regression problem by using tensor algebra and unfolding. For instance, with respect to mode 1 and using Equation 4.6, the NTD problem can be rewritten:

$$\mathcal{X} = \mathcal{G} \times_1 W \times_2 H \times_3 Q \Leftrightarrow \mathcal{X}_{(1)} = W\mathcal{G}_{(1)}(H \otimes Q)^\intercal. \tag{4.12}$$

2. We will see in Section 4.3.3 that the β-divergences generalize the Euclidean distance.

Hence, solving NTD with respect to W is equivalent to solving a matrix problem $\mathcal{X}_{(1)} \approx WV$ with $V = \mathcal{G}_{(1)}(H \otimes Q)^\intercal$ being fixed, *i.e.* equivalent to an NMF problem when fixing one of the factors.

> Thus, solving the NTD subproblem subject to one factor can be done by using algorithms designed to solve NMF in an alternating scheme.

This is of particular interest because NMF is largely studied in literature [Gil20], and many algorithms are designed to solve NMF by alternating between the factors [Gil20, Chap. 8]. Hence, in this thesis, each of the matrix factors W, H and Q are solved as matrix regression problems, using tools developed for NMF.

With respect to the core $\mathcal{G}$, the subproblem may be solved as an alternating vectorized problem (rank-1 NMF), using Equation 4.5:

$$\mathcal{X} = \mathcal{G} \times_1 W \times_2 H \times_3 Q \Leftrightarrow \text{vec}(\mathcal{X}) = (W \otimes H \otimes Q)\,\text{vec}(\mathcal{G}), \tag{4.13}$$

or as a tensor problem, using gradient descent [3]. Still, in practice, computing the Kronecker products ($H \otimes Q, W \otimes H \otimes Q$, etc) would be extremely inefficient, or even intractable, because it requires the computation of very large matrices [4]. In that sense, strategies are generally employed to avoid this computation, which are specific to the considered loss function, and are detailed in related sections (4.3.2 and 4.3.3).

The general framework to solve NTD with respect to β-divergences or the Euclidean distance is detailed in Algorithm 2. The term "arg reduce" refers to an optimization process decreasing the loss function, but which may not be the "arg min".

4.3.2 Euclidean-NTD: Optimizing the Euclidean Distance

One of the most common distance to solve optimization problems is the (squared) Euclidean distance d_{euc}. This distance, computed between two vectors, corresponds to the length of the straight line between these vectors seen as points in the associated Euclidean

3. Many more optimization strategies could probably be used, but this thesis restricts to both these methods.

4. For instance, given $W \in \mathbb{R}^{I \times I'}$, $H \in \mathbb{R}^{J \times J'}$ and $Q \in \mathbb{R}^{K \times K'}$, $W \otimes H \otimes Q \in \mathbb{R}^{IJK \times I'J'K'}$.

Algorithm 2: High-level organization of algorithms solving NTD.

Input: $\mathcal{X}$, initial values for $\mathcal{G}^{(0)}, W^{(0)}, H^{(0)}, Q^{(0)}$
Output: $\mathcal{G}, W, H, Q$
for $t = 1, 2, ...$ **do**
  $W^{(t+1)} = \underset{W \geq 0}{\arg \operatorname{reduce}} \quad d\left(\mathcal{X}_{(1)}, W \mathcal{G}_{(1)}^{(t)} (H^{(t)} \otimes Q^{(t)})^\intercal\right)$
  $H^{(t+1)} = \underset{H \geq 0}{\arg \operatorname{reduce}} \quad d\left(\mathcal{X}_{(2)}, H \mathcal{G}_{(2)}^{(t)} (W^{(t+1)} \otimes Q^{(t)})^\intercal\right)$
  $Q^{(t+1)} = \underset{Q \geq 0}{\arg \operatorname{reduce}} \quad d\left(\mathcal{X}_{(3)}, Q \mathcal{G}_{(3)}^{(t)} (W^{(t+1)} \otimes H^{(t+1)})^\intercal\right)$
  Updating $\mathcal{G}^{(t+1)}$ (details in sections 4.3.2 and 4.3.3)
end

space. Practically, for two vectors $x, y \in \mathbb{R}^L$:

$$d_{euc}(x, y) = \|x - y\|_2^2 = \sum_{l=1}^{L} (x_l - y_l)^2. \tag{4.14}$$

The Euclidean distance can be extended to matrices and tensors as the elementwise l_2 norm of the difference between two matrices/tensors, and is called the Frobenius norm, denoted as $\|.\|_F$. Hence, in the NTD framework, minimizing the squared Euclidean distance between the original tensor and the approximation defines the following optimization problem:

$$W^*, H^*, Q^*, \mathcal{G}^* = \underset{W \geq 0, H \geq 0, Q \geq 0, \mathcal{G} \geq 0}{\arg \min} \frac{1}{2} d_{Euc}(\mathcal{X}, \mathcal{G} \times_1 W \times_2 H \times_3 Q) \tag{4.15}$$

with $d_{Euc}(\mathcal{X}, \mathcal{G} \times_1 W \times_2 H \times_3 Q) = \sum_{i,j,k} (\mathcal{X}_{ijk} - (\mathcal{G} \times_1 W \times_2 H \times_3 Q)_{ijk})^2$. This optimization problem is called in this manuscript "Euclidean-NTD".

Many algorithms are designed to solve the Euclidean-NTD [PC11; Zho+15; Xu15; Mar+20]. Our particular way to solve the Euclidean-NTD [Mar+20] makes use of the Alternating Least Squares (ALS) method and, more precisely, a variant called Hierarchical ALS (HALS).

Nonnegative Least Squares problem

The Nonnegative Least Squares problem (NNLS) is a convex optimization problem, defined as:

$$\arg\min_{x \geq 0} \frac{1}{2} \|y - Ax\|_2^2, \tag{4.16}$$

i.e. $\arg\min_{x \geq 0} \frac{1}{2} d_{euc}(y, Ax)$ for a matrix A and a vector y. The NNLS problem can be solved exactly with active-set methods [LH95]. In the general formulation, A and y are not constrained to be nonnegative, but in our case both A and y are in general nonnegative.

The NNLS problem can be extended to matrices as: $\arg\min_{X \geq 0} \frac{1}{2} \|Y - AX\|_F^2$ because, in this formulation, X defines a concatenation of independent NNLS subproblems [NVG20]. The problem can be equivalently written as: $\arg\min_{X \geq 0} \frac{1}{2} \|Y^\intercal - X^\intercal A^\intercal\|_F^2$.

Finally, given $M \in \mathbb{R}^{m \times n}, V \in \mathbb{R}^{d \times n}$, and $d \in \mathbb{N}, d < \min(m, n)$, the NNLS problem is the problem of finding $U^* \in \mathbb{R}_+^{m \times d}$ such that:

$$U^* = \arg\min_{U \geq 0} \quad \frac{1}{2} \|M - UV\|_F^2. \tag{4.17}$$

Solving NTD and NMF as NNLS subproblems

Section 4.3.1 introduced the alternating strategy for solving NTD: fixing all factors except one, *e.g.* W, and solving the subproblem associated with this factor, *e.g.* $\mathcal{X}_{(1)} \approx WV$ with $V = \mathcal{G}_{(1)}(H \otimes Q)^\intercal$ being fixed. When using the Euclidean distance to optimize NTD, this subproblem is exactly a NNLS problem: $\arg\min_{W \geq 0} \frac{1}{2} \|\mathcal{X}_{(1)} - WV\|_F^2$. Hence, the NTD with respect to matrix factors W, H, Q can be solved using NNLS resolution methods. Analogously, NMF can be solved as NNLS subproblems.

In practice, both NMF and NTD being based on alternating schemes, it is not necessary to perform exact decompositions, which can be costly. Instead, it is sufficient to approximately solve each NNLS subproblem to guarantee the convergence of NMF to a stationary point [GG12], which is the sense of arg reduce.

Hierarchical Alternating Least Squares

Hierarchical Alternating Least Squares (HALS) is a method introduced by Cichocki *et al.* [CZA07] to approximately solve NNLS problems. The convergence of HALS is guaranteed as a particular case of the more general PALM optimization framework [BST14].

Instead of updating in one iteration the entire matrix U, HALS updates iteratively each column [5] $U_{:k}\,(1 \leq k \leq r)$. Hence, the subproblem: $U^* = \underset{U \geq 0}{\arg\min} \frac{1}{2} \|M - UV\|_F^2$ is itself divided in subproblems:

$$U_{:k}^* = \underset{U_{:k} \geq 0}{\arg\min} \frac{1}{2} \|M - \sum_{i \neq k} U_{:i} V_{i:} - U_{:k} V_{k:}\|_F^2. \tag{4.18}$$

Each columnwise subproblem admits an optimal closed-form solution, giving the following update rule [GG12]:

$$U_{:k}^{(t+1)} = \max\left(0, \frac{MV_{k:}^{(t)\intercal} - \sum_{i=0}^{k-1} U_{:i}^{(t+1)} V_{i:}^{(t)} V_{k:}^{(t)\intercal} - \sum_{i=k+1}^{r} U_{:i}^{(t)} V_{i:}^{(t)} V_{k:}^{(t)\intercal}}{V_{k:}^{(t)} V_{k:}^{(t)\intercal}}\right) \tag{4.19}$$

Denoting as $A = MV^{(t)\intercal}$ and $B = V^{(t)}V^{(t)\intercal}$, Equation 4.19 can be rewritten as: $U_{:k}^{(t+1)} = \max\left(0, \frac{A_{:k} - \sum_{i=0}^{k-1} U_{:i}^{(t+1)} B_{ik} - \sum_{i=k+1}^{r} U_{:i}^{(t)} B_{ik}}{B_{kk}}\right)$. Hence, A and B can be computed only once for the update of all columns $U_{:k}$ of U.

Accelerated variant of the HALS

In this work, we particularly implemented the accelerated variant of the HALS introduced by Gillis & Gilneur [GG12]. The idea of this variant is to make use of the fact that previously defined A and B matrices are computed only once for the update of all columns $U_{:k}^{(t)}$ of $U^{(t)}$ (at iteration t), and could then be reused to update $U^{(t)}$ several times in a row.

Indeed, updating $U^{(t)}$ several times instead of once before alternating to $V^{(t)}$ still results in only one computation of matrices A and B. The computation of these two matrices being in general the heaviest operation complexity-wise in the update rule, the second (and subsequent) updates for $U^{(t)}$ are relatively cheaper computationally than the first one.

Even if performing several updates for $U^{(t)}$ is itself more expensive than performing only one update, this technique leads to a better solution $U^{(t)}$ to the NNLS subproblem defined in Equation 4.17 at a relatively cheap cost, which, in the end, could reduce the overall number of iterations t.

Results presented in [GG12] indeed show an overall acceleration of the accelerated

5. This is the sense of "Hierarchical".

variant of HALS over the traditional one (where each factor is updated once before alternating), without losing the convergence properties.

Practically, this accelerated variant results in the following routine:

— 1- Compute A and B for the update t of $U^{(t)}$,

— 2- Update each column $U^{(t)}_{:k}$ of $U^{(t)}$,

— 3- Repeat step 2 several times (this is called "inner iterations", denoted as $U^{(t,l)}$),

— 4- Alternate factors, and perform analogously for $V^{(t)}$.

Finally, step 3 needs the definition of a stopping criterion to halt the inner iterations, or at least a definition of "several times" (how many inner iterations l should be performed for $U^{(t,l)}$ before alternating to $V^{(t)}$). This can be achieved in three different ways [GG12]:

— The first way estimates the gain ρ between the first and the subsequent updates (*i.e.* the gain between computing A and B and reusing them), and then fixes the number of inner iterations as $\lfloor 1 + \alpha\rho \rfloor$, α being an hyperparameter. Following the experimental results of [GG12], we fix $\alpha = 0.5$.

In theory, ρ is computed as the ratio between the computational complexity of the first and the second update. In practice, instead of computing the exact computational complexities, a cheaper way (but not a deterministic one) consists of measuring the time of both updates.

— The second way consists of evaluating the gain in reconstruction error of these inner iterations relatively to the NNLS subproblem (Equation 4.17), and halt the inner iterations if the improvement becomes too small.

In practice, this can be achieved by computing the initial error $\|U^{(t,1)} - U^{(t,0)}\|_F^2$, and halting the inner iterations at l when $\|U^{(t,l)} - U^{(t,l-1)}\|_F^2 \leq \eta \|U^{(t,1)} - U^{(t,0)}\|_F^2$. Hyperparameter η is fixed to 0.01, meaning that inner iterations stop when the current gain in reconstruction error becomes 100 times lower than the gain between the first two inner iterations.

— The third way is an absolute maximal number of inner iterations.

In practice, we implemented all of these stopping criteria. The final algorithm is presented in Algorithm 3.

Core Update

The core update could be carried with the same HALS method, using Equation 4.5, as: $\arg\min_{\mathcal{G} \geq 0} \frac{1}{2} \| \operatorname{vec}(\mathcal{X}) - (W \otimes H \otimes Q) \operatorname{vec}(\mathcal{G}) \|_2^2$ is a vectorial NNLS problem. Still, this solution requires the computation of the Kronecker product $W \otimes H \otimes Q$, which is generally costly,

Algorithm 3: One step of Accelerated HALS on $U^{(t)}$

Input: $M \in \mathbb{R}_+^{n \times m}, U^{(t)} \in \mathbb{R}_+^{n \times r}, V^{(t)} \in \mathbb{R}_+^{r \times m}, \alpha, \eta, \text{maxiter}$
Output: $U^{(t+1)}$
$A = MV^{(t)\intercal}, B = V^{(t)}V^{(t)\intercal}$
for $k = 1, 2, ..., r$ **do**
  if $B_{kk} \neq 0$ **then**

$$U_{:k}^{(t+1,0)} = \max\left(0, \frac{A_{:k} - \sum_{i=0}^{k-1} U_{:i}^{(t+1,0)} B_{ik} - \sum_{i=k+1}^{r} U_{:i}^{(t)} B_{ik}}{B_{kk}}\right)$$

for $k = 1, 2, ..., r$ **do**
  if $B_{kk} \neq 0$ **then**

$$U_{:k}^{(t+1,1)} = \max\left(0, \frac{A_{:k} - \sum_{i=0}^{k-1} U_{:i}^{(t+1,1)} B_{ik} - \sum_{i=k+1}^{r} U_{:i}^{(t+1,0)} B_{ik}}{B_{kk}}\right)$$

$\rho = \frac{\text{time}(U_{:k}^{(t+1,0)})}{\text{time}(U_{:k}^{(t+1,1)})}$
`/* The time for` $U_{:k}^{(t+1,0)}$ `includes the computation of` A `and` B `*/`
for $l = 1, 2, ..., \min(\lfloor 1 + \alpha\rho \rfloor, maxiter)$ **do**
  for $k = 1, 2, ..., R$ **do**
    if $B_{kk} \neq 0$ **then**

$$U_{:k}^{(t+1,l+1)} = \max\left(0, \frac{A_{:k} - \sum_{i=0}^{k-1} U_{:i}^{(t+1,l+1)} B_{ik} - \sum_{i=k+1}^{r} U_{:i}^{(t+1,l)} B_{ik}}{B_{kk}}\right)$$

  if $\|U^{(t,l+1)} - U^{(t,l)}\|_F \leq \eta \|U^{(t,1)} - U^{(t,0)}\|_F$ **then**
    break.

with a complexity of $\mathcal{O}(IJKI'J'K')$. In practice, computing this product can be inefficient or even intractable for large dimensions.

Computational complexity can be reduced by using the multi-mode product instead (*i.e.* n-mode product on several modes), using Equation 4.5. Still, the vectorization operation: $\text{vec}(\mathcal{G} \times_1 W \times_2 H \times_3 Q)$ still requires the computation of large vectors, and HALS applied on vectors reduces to the update of each coefficient independently, which can be costly.

Instead, we choose to solve the subproblem: $\arg\min_{\mathcal{G} \geq 0} \frac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 W \times_2 H \times_3 Q\|_2^2$ with a proximal gradient descent, *i.e.* $\mathcal{G}^{(t+1)} = \max(0, \mathcal{G}^{(t)} - \gamma \nabla \mathcal{G}^{(t)})$. Using Equations 4.3 and 4.5,

the[6] gradient $\nabla\mathcal{G}$ is equal to:

$$\nabla\mathcal{G} = \mathcal{G} \times_1 W^\intercal W \times_2 H^\intercal H \times_3 Q^\intercal Q - \mathcal{X} \times_1 W^\intercal \times_2 H^\intercal \times_3 Q^\intercal. \tag{4.20}$$

The gradient step γ is (globally) optimal when equal to the inverse of the product of the largest singular values of $W^\intercal W$, $H^\intercal H$ and $Q^\intercal Q$ [Bec17, Chap. 10], *i.e.* denoting as λ_W, λ_H and λ_Q the largest singular values of $W^\intercal W$, $H^\intercal H$ and $Q^\intercal Q$ respectively, $\gamma = \frac{1}{\lambda_W \lambda_H \lambda_Q}$.

4.3.3 β-NTD: Optimizing the β-divergences

The family of β-divergences was introduced in [Bas+98]. Given two nonnegative scalars x and y, the β-divergence between x and y, denoted as $d_\beta(x|y)$, is defined as follows:

$$d_\beta(x|y) = \begin{cases} \frac{x}{y} - \log(\frac{x}{y}) - 1 & \beta = 0 \\ x\log(\frac{x}{y}) + (y - x) & \beta = 1 \\ \frac{x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}}{\beta(\beta-1)} & \beta \in \mathbb{R}\backslash\{0,1\} \end{cases} \tag{4.21}$$

These divergences generalize the Euclidean distance ($\beta = 2$), the Kullback-Leibler (KL) divergence ($\beta = 1$) and the Itakura-Saito (IS) divergence ($\beta = 0$). This family of divergences can be of particular interest in the presence of specific noise distributions, such as KL for i.i.d. Poisson noise [LS99], or IS for i.i.d. multiplicative Gamma noise [FBD09], see [Gil20, Chap. 5] for a quick overview.

The β-divergences $d_\beta(x|y)$ are homogeneous of degree β, that is for any $\lambda \in \mathbb{R}$:

$$d_\beta(\lambda x|\lambda y) = \lambda^\beta d_\beta(x|y). \tag{4.22}$$

It implies that factorizations obtained with $\beta > 0$ (such as the Euclidean distance or the KL divergence) rely more heavily on the largest data values and less precision is to be expected in the estimation of the low-power components. The IS divergence ($\beta = 0$) is scale-invariant and is the only one in the family of β-divergences to possess this property. It implies that low power entries are as important in the divergence computation as high-power areas. For instance, approximating as 10 an entry equal to 1 results in the same IS divergence error than approximating as 10^{-4} an entry equal to 10^{-5}.

More generally, the lower is β, the more impact is given to the good fit of low-power coefficients in the error computation. This property may be particularly useful for the

6. We drop the iteration notation $.^{(t)}$ for clarity.

processing of audio signals. Indeed (as presented in Section 2.2.4), low-frequency STFT and Mel coefficients are generally observed to be larger in magnitude than high-frequency coefficients. In addition, in audio signals, components of low intensity can perceptually be as important as components of high intensity, due to the perception of loudness in the human ear.

> In audio signals, using divergences which rely less on the largest coefficients and more on low-power coefficients can better suit the particular characteristics of audio signals, as components of low intensity can perceptually be as important as components of high intensity.

In particular, both KL and IS divergences are notoriously known to be better suited to audio source separation than the Euclidean distance [FBD09]. An overview of works using different β values in the audio domain ($\beta \in \{0, 0.5, 1\}$ in general) can be found in the thesis of Lefèvre [Lef12, Chap. 2].

This section focuses on the computation of a candidate solution to approximate NTD with β-divergence as a loss function:

$$W^*, H^*, Q^*, \mathcal{G}^* = \underset{W\geq 0, H\geq 0, Q\geq 0, \mathcal{G}\geq 0}{\arg\min} d_\beta(\mathcal{X}|\mathcal{G} \times_1 W \times_2 H \times_3 Q) \tag{4.23}$$

with d_β the elementwise β-divergence between two tensors.

Similarly than for the Euclidean distance, the loss function is non-convex with respect to all factors, and computing a global solution to NTD is NP-Hard [Vav10] (since NTD is a generalization of NMF). However, each subproblem obtained when fixing all but one mode is convex as long as $\beta \in [1, 2]$, and remains solvable for other β values [Gil20].

Hence, while recent work [MGF21] focused on jointly optimizing the problem with respect to both factors "at-once", alternating schemes are still standard to solve NMF with respect to β-divergences [LS99; FI11; Gil20]. In particular, the seminal paper by Lee and Seung [LS99] proposed an alternating algorithm for NMF with respect to the KL-divergence based on the multiplicative updates (MU) rule, later revisited and extended to β-divergences by Févotte and Idier [FI11]. We further extend the MU rule for NTD [Mar+22].

Multiplicative Updates rule for Alternating NMF

Recalling that we denote as $\cdot$ and $\div$ the element-wise product and division, the multiplicative updates (MU) rule in approximate NMF ($M \approx UV^\intercal$) is defined as:

$$U^{(t+1)} = \max\left(U^{(t)} \cdot \left(\frac{\left[(U^{(t)}V^{(t)})^{\cdot(\beta-2)} \cdot M\right] V^{(t)\intercal}}{(U^{(t)}V^{(t)})^{\cdot(\beta-1)}V^{(t)\intercal}}\right)^{\cdot\gamma(\beta)}, \epsilon\right) . \tag{4.24}$$

In Equation 4.24, $\epsilon > 0$ is a small constant. The element-wise maximum between the matrix update (*i.e.* the closed-form expression of the minimizer of the majorization built at the current iterate) and ϵ in Equation 4.24 aims at avoiding zero entries in factors and establishing convergence guarantee to stationary points within the BSUM framework [RHL13]. Zero entries in factors may cause division by zero and the zero-locking phenomenon.

Function $\gamma(\beta)$ in Equation 4.24 is a function such that [FI11]:

$$\gamma(\beta) = \begin{cases} \frac{1}{2-\beta} & \beta < 1 \\ 1 & 1 \leq \beta \leq 2 \\ \frac{1}{\beta-1} & \beta > 2 \end{cases} \tag{4.25}$$

MU for NTD

The previous MU rule can be used to solve the different alternating subproblems presented in Algorithm 2. A difficulty is that forming the Kronecker products is bound to be extremely inefficient both in terms of memory allocation and computation time.

Instead, in the MU rule for updating the factor matrix W, the matrix $V = \mathcal{G}_{(i)}\,(H \otimes Q)^\intercal$ can be computed efficiently using Equation 4.6 (setting A as the identity matrix):

$$\mathcal{G}_{(1)}\,(H \otimes Q)^\intercal = (\mathcal{G} \times_2 H \times_3 Q)_{(1)}, \tag{4.26}$$

which brings down the complexity of forming V from $\mathcal{O}(KLJ'K'L')$ if done naively to $\mathcal{O}(KJ'K'L' + LJ'KL')$, and drastically reduces memory requirements. The same rule is used for H and Q.

For the core factor, one can use the vectorization property of Equation 4.5 (*i.e.* using $\mathcal{X} - \mathcal{G} \times_1 W \times_2 H \times_3 Q = \text{vec}(\mathcal{X}) - (W \otimes H \otimes Q)\,\text{vec}(\mathcal{G})$) to relate the core update with the NMF MU rules. Once again, matrix $U = W \otimes H \otimes Q$ is $J'K'L'$ times larger than the

data itself. Therefore, to avoid computing heavy Kronecker products, we use Equation 4.5, which states that $(W \otimes H \otimes Q)\,\mathrm{vec}(\mathcal{G}) = \mathrm{vec}(\mathcal{G} \times_1 W \times_2 H \times_3 Q)$.

This results in Algorithm 4, presenting one loop of update for the NTD, with respect to the β-divergence.

Algorithm 4: A loop of β_NTD($\mathcal{X}$,dimensions,β)

Input: $\mathcal{X}, \mathcal{G}^{(t)}, W^{(t)}, H^{(t)}, Q^{(t)}, \epsilon, \beta$
Output: $\mathcal{G}^{(t+1)}, W^{(t+1)}, H^{(t+1)}, Q^{(t+1)}$

$V_W = (\mathcal{G}^{(t)} \times_2 H^{(t)} \times_3 Q^{(t)})_{(1)}$

$W^{(t+1)} = \max\left(W^{(t)} \cdot \left(\frac{\left[(W^{(t)}V_W)^{\cdot(\beta-2)} \cdot \mathcal{X}_{(1)}\right]V_W^\intercal}{(W^{(t)}V_W)^{\cdot(\beta-1)}V_W^\intercal}\right)^{\cdot\gamma(\beta)}, \epsilon\right)$

$V_H = (\mathcal{G}^{(t)} \times_2 W^{(t+1)} \times_3 Q^{(t)})_{(1)}$

$H^{(t+1)} = \max\left(H^{(t)} \cdot \left(\frac{\left[(H^{(t)}V_H)^{\cdot(\beta-2)} \cdot \mathcal{X}_{(2)}\right]V_H^\intercal}{(H^{(t)}V_H)^{\cdot(\beta-1)}V_H^\intercal}\right)^{\cdot\gamma(\beta)}, \epsilon\right)$

$V_Q = (\mathcal{G}^{(t)} \times_2 W^{(t+1)} \times_3 H^{(t+1)})_{(1)}$

$Q^{(t+1)} = \max\left(Q^{(t)} \cdot \left(\frac{\left[(Q^{(t)}V_Q)^{\cdot(\beta-2)} \cdot \mathcal{X}_{(3)}\right]V_Q^\intercal}{(Q^{(t)}V_Q)^{\cdot(\beta-1)}V_Q^\intercal}\right)^{\cdot\gamma(\beta)}, \epsilon\right)$

$\mathcal{N} = (\mathcal{G}^{(t)} \times_1 W^{(t+1)} \times_2 H^{(t+1)} \times_3 Q^{(t+1)})^{\cdot(\beta-2)} \cdot \mathcal{X}$

$\mathcal{D} = (\mathcal{G}^{(t)} \times_1 W^{(t+1)} \times_2 H^{(t+1)} \times_3 Q^{(t+1)})^{\cdot(\beta-1)}$

$\mathcal{G}^{(t+1)} = \max\left(\mathcal{G}^{(t)} \cdot \left(\frac{\mathcal{N} \times_1 W^{(t+1)\intercal} \times_2 H^{(t+1)\intercal} \times_3 Q^{(t+1)\intercal}}{\mathcal{D} \times_1 W^{(t+1)\intercal} \times_2 H^{(t+1)\intercal} \times_3 Q^{(t+1)\intercal}}\right)^{\cdot\gamma(\beta)}, \epsilon\right)$

A Note on MU for Euclidean Distance

The 2-divergence (β-divergence with $\beta = 2$) is exactly the Euclidean distance. Hence, the previous algorithm with MU rule is also suited to solve NTD with respect to the Euclidean distance. In practice though, HALS is known to be more efficient than MU [GG12]. In that sense, only HALS is considered to solve NTD with respect to the Euclidean distance.

4.4 Musical Barwise Interpretation

The previous sections were dedicated to the algebraic and algorithmic aspects of NTD (respectively Sections 4.2 and 4.3). The current section is dedicated to its interpretation as a barwise factorization technique for music. Firstly, we introduce how a spectrogram can be folded into a tensor, called "Time-Frequency-Bar" tensor, with two temporal modes,

and, secondly, we propose a tentative interpretation for the different factors W, H, Q and $\mathcal{G}$.

4.4.1 TFB Tensor

NTD is a tensor decomposition model. In that sense, it must be applied on a 3$^{\text{rd}}$-dimensional representation of music, which we introduce under the name of "TFB tensor", standing for "Time-Frequency-Bar tensor".

Music in its audio form is often represented in the time-frequency domain as a spectrogram, *i.e.* a 2-dimensional matrix (further denoted as X). Along the x-axis, the temporal dimension unfolds, with the indices of signal frames, while the y-axis is a frequency-related index (indices of Fourier coefficients, pitches, wavelet coefficients, etc).

In the TFB representation, the temporal dimension is broken up into two distinct dimensions: a dimension for time at a low-scale, representing time in each bar, and a barwise dimension. This result in a tensor with a frequential mode, a mode for time at barscale, and a mode for the different bars, represented in Figure 4.4.

Analogously to the Barwise TF matrix, introduced in Section 2.4.2, time at barscale is re-sampled in S values, where S is the subdivision parameter.

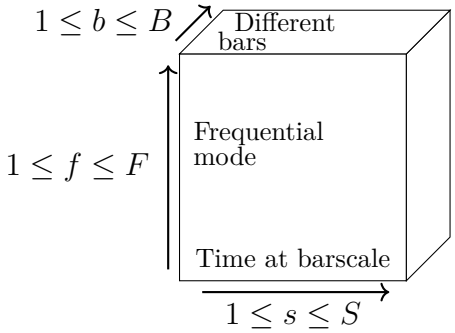


Figure 4.4 – TFB tensor

NTD is then applied to the TFB tensor, in any feature representation (STFT, Mel, chromagram, ...). In that paradigm, each factor matrix relates to one particular mode: W relates to the frequential mode, H to the inner bar time, and Q to the mode indexing the different bars, and the core $\mathcal{G}$ mix these different factors.

This section aims at explaining what each of these factors represents in the barwise musical context.

4.4.2 Factors Interpretation

For the NTD applied on the TFB tensor $\mathcal{X} \in \mathbb{R}_+^{F \times S \times B}$, NTD core dimensions are denoted as F', S' and B'. Hence, $W \in \mathbb{R}_+^{F \times F'}$, $H \in \mathbb{R}_+^{S \times S'}$, $Q \in \mathbb{R}_+^{B \times B'}$ and $\mathcal{G} \in \mathbb{R}_+^{F' \times S' \times B'}$.

NMF for Music Transcription

Let's start with W, by taking the example of NMF for Music Transcription (NMF was presented in Section 4.2.3). It consists of factorizing a nonnegative matrix $X \in \mathbb{R}_+^{M \times N}$ (here, a spectrogram) in two nonnegative matrices W and V such that $X \approx WV = \sum_{d=1}^{D} W_{:d} V_{d:}$. NMF has been extensively used in audio signal processing, generally for source separation [Vir07; OF09; Lef12] and music transcription [SB03; VBB09; Che+16; Gao+17; WMC22].

The task of music transcription aims at representing the musical content of an audio signal into a symbolic format (typically a musical score), *i.e.* converting spectrograms into frequential information (which note is played?) and temporal information (when is this note played?). With NMF, W represents frequential information while V represents temporal information. A toy example is represented in Figure 4.5, adapted from [WMC22].

In this example, each column $W_{:d}$ of W represents the frequential content of a note, and each associated row $V_{d:}$ represents the activations of this particular note. Hence, W is a concatenation of the frequential content of several notes (here, 3 columns for G, A and B), representing templates of frequential content.

In NTD, W holds the same purpose. Matrix V is used in transcription to indicate when notes are played, which is not transferable as such in the NTD framework.

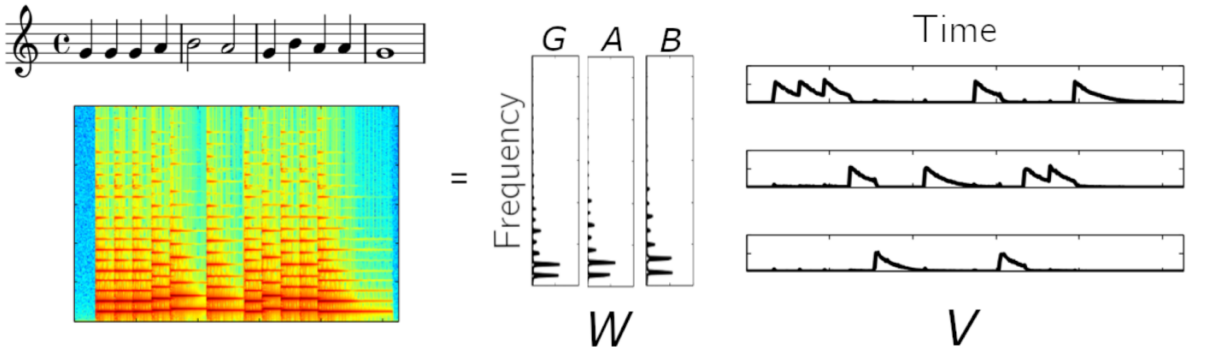


Figure 4.5 – A toy example of transcription using NMF, adapted from [WMC22].

Nonnegative Matrix Tri-Factorization

In the NMF model, each column of W is linked to a unique row in V by conventional matrix product. This model can be extended by adding a "mix matrix" G, such that a spectrogram X is decomposed in three nonnegative matrices[7] $W \in \mathbb{R}_+^{M \times D_W}, G \in \mathbb{R}_+^{D_W \times D_H}, H \in \mathbb{R}_+^{D_H \times N}$ such that $X = WGH$.

This model is referred to in the literature as Nonnegative Matrix Tri-Factorization (NMTF) [Din+06] or three-factor NMF [Cic+09]. Pushing the analogy with NMF, the idea of NMTF is to represent X as conic combinations of columns of W with rows of H, defined in the matrix G. Practically, $X \approx \sum_{d_W=1}^{D_W} \sum_{d_H=1}^{D_H} G_{d_W d_H} W_{:d_W} H_{d_H:}$, meaning that coefficient (d_W, d_H) of G is the weight of the combination of $W_{:d_W}$ and $H_{d_H:}$. NMF is a particular case of NMTF where $G = I_{D_W}$ and $D_W = D_H$.

Applied to a barwise spectrogram X, we interpret NMTF as modeling the musical content in X as a mix between frequential templates in W and barwise rhythmic templates in H. This notion of "barwise rhythmic template" is novel with respect to the traditional interpretation of NMF in audio domain, but seems rather clear in music composition.

In conventional music notation, rhythm is represented with symbols such as "quarter notes", "rests" and "beamed notes", as presented in Section 2.2.2. A barwise rhythmic template is formed of a group of symbols, regularly spaced at the bar scale, representing the onsets of musical events in this bar. For example, in a $\frac{4}{4}$ metric, an event occurring on each beat is represented with 4 consecutive quarter notes.

7. Note that we changed the notation of the previous matrix V by matrix H to disambiguate the different roles of these matrices between NMF and NTD, both relating to time information.

Hence, for a barwise spectrogram, NMTF associates frequential templates (*e.g.* musical notes) with rhythmic templates at the barscale (*e.g.* each beat in the bar), forming musical content at the barscale. Frequential and rhythmic templates can be interpreted as encoding respectively the pitches and durations of the different notes.

As an example, we study a simple drum pattern, presented in symbolic format in Figure 4.6. This drum pattern can be factorized with NMF or NMTF. An example of decompositions for both models is presented in Figure 4.7.

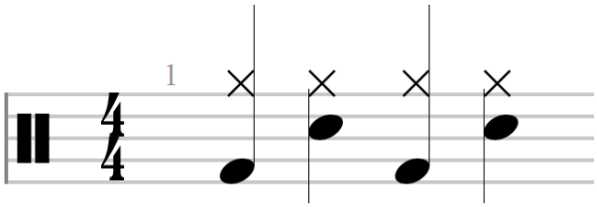


Figure 4.6 – A simple drum pattern, taken as an example for NMTF

Musical Pattern

Mathematically, unconstrained NMTF can always be recast as an NMF [Din+06], typically by setting $V = GH$. The relevance of NMTF becomes more obvious when considering different matrices $(G^{(b')})_{b' \in \mathbb{N}}$ applying on the same W and H matrices.

In this context, the different "mix matrices" $G^{(b')}$ define different combinations between the columns of W and of H. When considering W and H as dictionaries, each $G^{(b')}$ defines a (potentially sparse) dictionary combination, *i.e.* a combination of frequential and barwise rhythmic templates, resulting in a barwise spectrogram. The concatenation of different matrices $G^{(b')}$ results in a tensor $\mathcal{G}$.

Musical pattern

We call **"musical pattern"**, denoted as $P_{b'}$, the product $W\mathcal{G}_{::b'}H^\intercal$, which corresponds to a barwise spectrogram resulting of a combination of frequential and rhythmic templates.

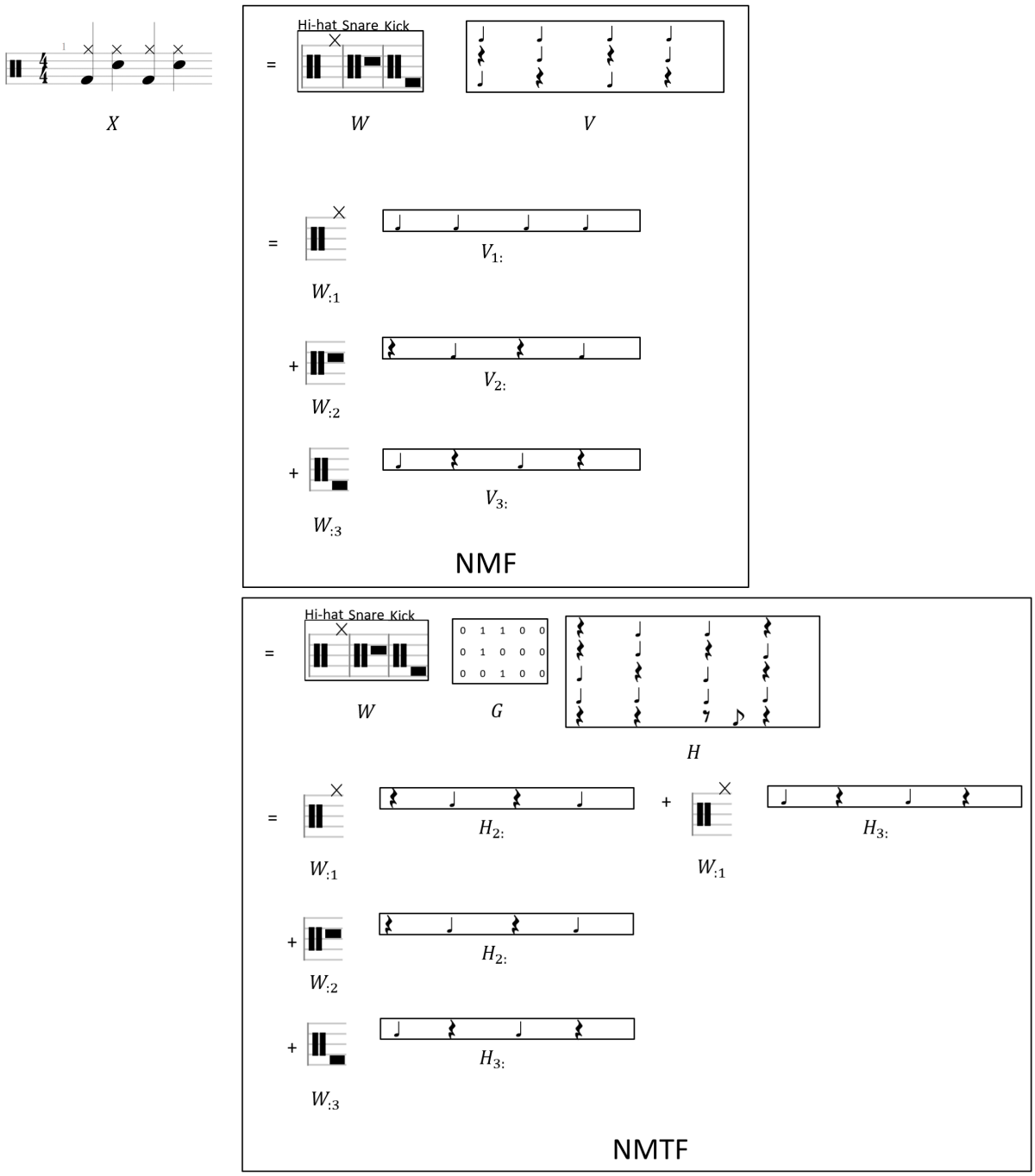


Figure 4.7 – Drum pattern from Figure 4.6, decomposed in both NMF and NMTF models. Matrix H is composed of 5 different barwise rhythmic templates, but only 2 are actually used ($H_{2:}$ and $H_{3:}$), presenting H as a dictionary of rhythmic patterns. This is useful in the interpretation of NTD, as matrices W and H can be interpreted as dictionaries of pitch-related and duration-related information (at barscale) in the song.

NTD: Musical Patterns as Barwise Features

Finally, NTD consists of factorizing a TFB tensor $\mathcal{X}$ in a product $\mathcal{G} \times_1 W \times_2 H \times_3 Q$. In this decomposition, W consists of frequential templates, H in barwise rhythmic templates, and the different slices of $\mathcal{G}$ consist of mix matrices between these dictionaries.

Matrix Q represents the barwise information of the original content $\mathcal{X}$, which is missing from the other factors. In particular, any given bar of index b in the TFB tensor is represented as:

$$\mathcal{X}_{::b} \approx W \left(\sum_{b'=1}^{B'} Q_{bb'} \mathcal{G}_{::b'} \right) H^\intercal = \sum_{b'=1}^{B'} Q_{bb'} P_{b'}, \tag{4.27}$$

i.e. a conic combination of musical patterns.

> Matrix Q represents the different bars in the song from the musical patterns. Hence, musical patterns can be interpreted as barwise features in the Q matrix.

Figure 4.8 presents a practical example when using NTD on the song *Come Together* by The Beatles, represented as a chromagram. NTD is computed with dimensions $F' = 12, S' = 12, B' = 10$.

Chromagrams (as presented in Section 2.2.4) are features where the frequential content is represented in the chromatic scale, corresponding to each of the 12 semi-tones. As a consequence, we do not expect factorization on this mode: notes are already expressed by different columns and 12 is a small dimension compared to the other modes. In addition, it seems detrimental to lower the dimensionality compared to the drawback of factorizing notes (*i.e.* assuming that some notes do not appear in the entire song or only appear in a same chord). Hence, when processing chromagrams, W is fixed to I_{12}, *i.e.* the 12-size identity matrix, as in [Mar+20].

In the end, structural segmentation is studied through the Q matrix, and the musical patterns $P_{b'} = W \mathcal{G}_{::b'} H^\intercal$, $\forall 1 \leq b' \leq B'$ are studied in a pattern uncovering task.

Robustness of the Interpretation

A tentative interpretation for the factors in the decomposition is presented in the current section, which seems valid in the particular example shown in Figure 4.8. In theory though, there might be several solutions $W, H, Q, \mathcal{G}$ that provide the same (or a very similar) estimate $\mathcal{G} \times_1 W \times_2 H \times_3 Q \approx \mathcal{X}$.

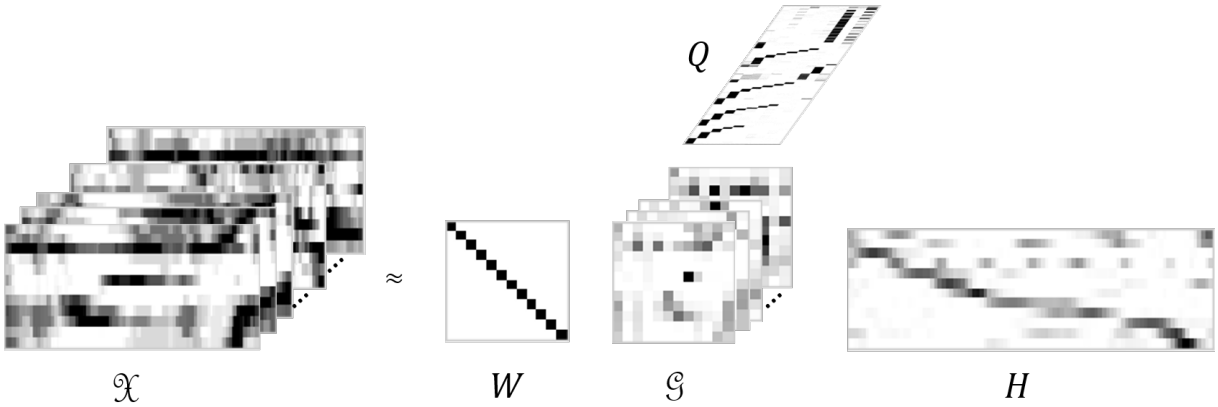


Figure 4.8 – NTD example on the chromagram of *Come Together* by The Beatles, with dimensions $F' = 12, S' = 12, B' = 10$, and $W = I_{12}$. Columns of the H and Q matrices (and, accordingly, slices of $\mathcal{G}$) are reordered for visualization purposes.

This problem, known as identifiability deficiency, has been little studied for NTD, and established identifiability conditions are very restrictive [Zho+15]. Moreover, these conditions are hard to check in practice [8]. Therefore it is unreasonable to assess the identifiability of the NTD in our application. As a consequence, this means that there might be infinitely many solutions minimizing Equation 4.9 that are, from an optimization point of view, equally satisfying.

Secondly, even in the case where the NTD is identifiable, the loss function 4.9 is highly non-convex, and local algorithms can only hope to recover a local minimum at best.

These two issues combined give rise to a high dependency of the solution on the initial condition: from two different initializations, two different results are likely to be obtained, most probably non-identifiable local minima. We have observed such situations in our investigations, with various initializations indeed resulting in different outputs.

However, in practice, the decomposition provides results that are reasonably interpretable from a musical perspective, as we present in Section 4.5. In particular, we initialized the algorithm with the absolute values of the Higher Order SVD [DDV00] computed with the *Tensorly* toolbox [Kos+19], resulting in a consistent and deterministic initialization.

Nevertheless, perfectly controlled convergence of the NTD decomposition on music

8. For instance, one can show that the identifiability of the parameters is conditioned on the identifiability of the Nonnegative Matrix Factorizations obtained by unfolding the tensor $\widehat{\mathcal{X}}$ along each dimension [Zho+15]

data remains a challenging research topic. Finding appropriate priors or additional constraints to disambiguate the outputs of the decomposition is an open problem, and conditions that may be specific to music signal processing should be investigated. In particular, in most NTD works, sparsity constraints are set on the core [MHA08].

A Note on Factor Normalization In addition to the identifiability deficiency, the decomposition is ambiguous with respect to scaling factors, *e.g.* $\mathcal{G} \times_1 W \times_2 H \times_3 Q = \mathcal{G} \times_1 (\lambda W) \times_2 (\frac{1}{\lambda} H) \times_3 Q$ for any $\lambda \in \mathbb{R}_+^*$. As a convention, the decomposition is normalized for every factor except Q, in order to let the energy discrepancies between the different bars appear in Q.

In practice, normalization means that every column $W_{:f'}$ and $H_{:s'}$, and every slice $\mathcal{G}_{::b'}$ is divided by its l_2 norm. This choice was motivated in the very beginning because Q serves for structural segmentation (see future Section 4.5.1), and we have assumed that energy discrepancies could help disambiguate the structure. Still, we have not made experiments to confirm this assumption, and other normalization paradigms could be employed (such as normalizing the matrix factors and not the core).

Dimensioning NTD

Dimensions F', S' and B' (dimensions of the core, number of columns in each factor, one for each mode) are crucial parameters of NTD. Indeed, low values for F', S' and B' tend to over-compress information in the data, failing to uncover relevant patterns in the song. Conversely, high values for F', S' and B' may give too much importance to details in the data, resulting in the inability of the model to extract relevant structural information by over-specializing the different musical patterns.

As developed further in Section 4.5, our experiments indicate that the optimal dimensions for F', S' and B' are probably specific for each song, which can be easily understood as a consequence of the diversity of intrinsic variability across music pieces. Providing an efficient and accurate method for selecting the dimensions is a challenging topic, left to future work. It is certainly another aspect that deserves further research (including its link with hierarchical levels of music description).

4.5 Experiments

Finally, this section presents experimental frameworks to evaluate NTD on real musical applications. In particular, this section studies the structural segmentation [Mar+20; Mar+22] and pattern uncovering tasks [SG18; Mar+22].

For structural segmentation, this section presents how NTD, by computing barwise representation of a music, can serve to compute barwise similarities, finally fueled to the CBM algorithm presented in Chapter 3. Pattern uncovering consists of a qualitative evaluation of musical patterns $P_{b'}$.

We compare 3 methods in these experiments: the Euclidean-NTD, optimizing the Euclidean distance with HALS, and both KL- and IS-NTD, optimizing the Kullback-Leibler ($\beta = 1$) and Itakura-Saito ($\beta = 0$) divergences with MU.

4.5.1 NTD for Structural Segmentation

NTD serves as barwise dimensionality reduction when applied to a collection of barwise spectrograms. In particular, matrix Q results in a barwise feature representation of the song, where features are the activations of the musical patterns defined by other factors.

As a consequence, Q can be used to compute autosimilarity matrices, such as $A_{cos}(Q)_{ij} = \frac{\langle Q_i, Q_j \rangle}{\|Q_i\|_2 \|Q_j\|_2} = \sum_{k=1}^{b'} \tilde{Q}_{ik}\tilde{Q}_{jk}$, *i.e.* the normalized dot product between each barwise representation (with musical patterns as features). Figure 4.9 compares the autosimilarities computed on the Barwise TF matrix with the NTD-based autosimilarities.

Segmentation can then be obtained by using the CBM algorithm on these autosimilarities. In these experiments, the core dimensions take their values in $\{8, 16, 24, 32, 40\}$. Due to the high number of resulting configurations (for each song, $5^3 = 125$ decompositions), we restrict this part to the study of RWC Pop, which is a smaller dataset than SALAMI. As an example, on an Intel® Core(TM) i7 CPU, decomposing the song *POP01* with dimensions $F' = 16, S' = 16, B' = 16$ for the Nonnegative Log Mel (NNLM) spectrogram takes approximately 40 seconds for the Euclidean-NTD, and respectively 2 $\frac{1}{2}$ minutes and 5 $\frac{1}{2}$ minutes for KL- and IS-NTD.

Following experimental conditions defined in Appendix A.2, the core dimensions are fitted by cross-validation: the RWC Pop dataset is divided in two subsets, songs with odd and even indexes, which alternatively act as test subset while the other half is used to set the optimal values for the core dimensions.

Parameter λ in the CBM algorithm (weighting the penalty function) is fixed to 1

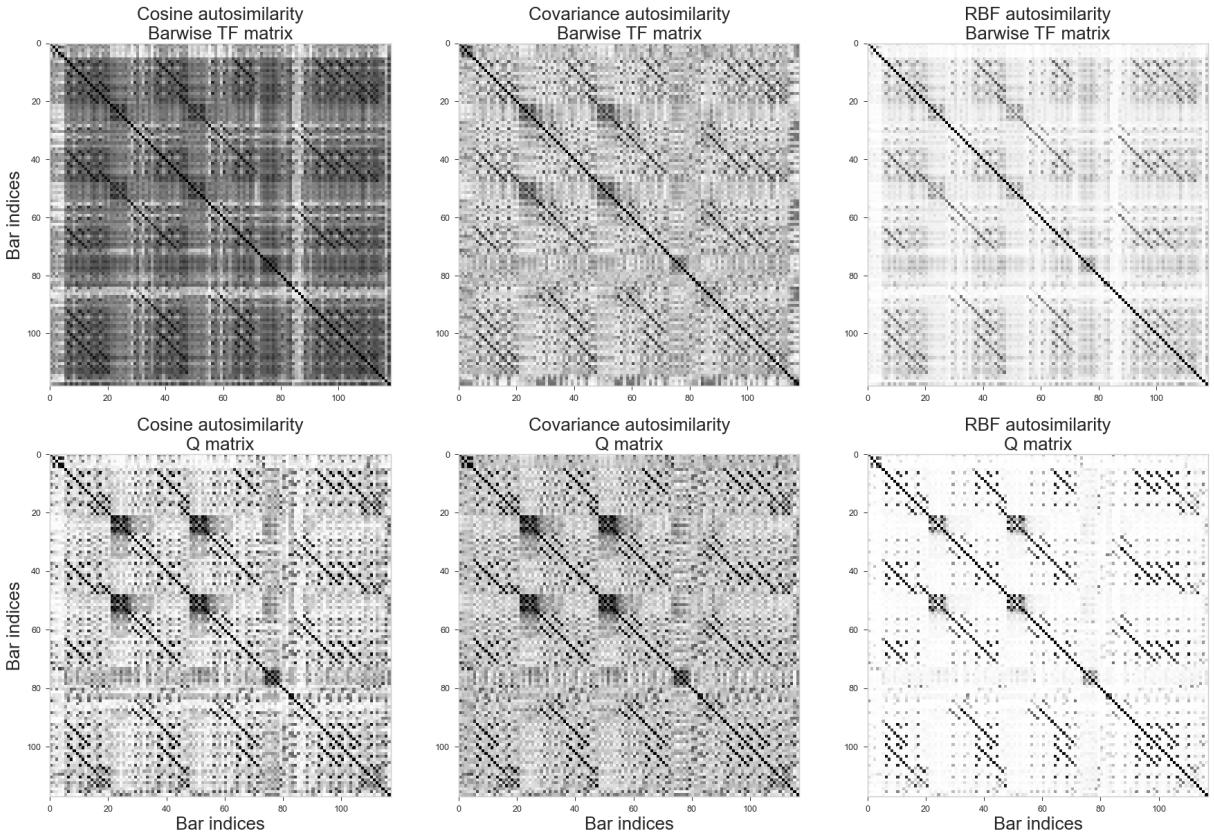


Figure 4.9 – Different autosimilarities on the song *POP01* of RWC Pop, computed on the feature (Barwise TF matrix, top) and on the Q matrix resulting of the NTD (bottom). Our particular example for NTD was taken for the chromagram feature, with Euclidean-NTD and by setting $W = I_{12}, S' = 16$ and $B' = 16$.

instead of being learned by cross-validation, in order to reduce the number of parameters in cross-validation and reduce the computational complexity of the experimental phase. Some experiments on early work (not presented here) showed that λ was of little impact on the NTD-based results.

The remainder of this section aims at comparing the structural segmentation performance obtained with NTD-based autosimilarities with the performance obtained with Barwise TF autosimilarities. Hence, experiments compare the different similarity functions (Cosine, Covariance and RBF) and the different features.

NTD being a nonnegative model, only nonnegative features are relevant as inputs. In that sense, we study songs represented as chromagrams, Mel and NNLM spectrograms. In addition, we compare Euclidean-, KL- and IS-NTD, studying the impact of the loss function with respect to the different features.

This set of experiments requires $3 \times 3 \times 3 = 27$ distinct conditions, which may be hard to test exhaustively. In order to focus conclusions, we design experiments trying to answer the three following questions:

Question 4 *With the current version of the CBM algorithm applied to NTD-based autosimilarities, how are segmentation performance impacted by the similarity function (Cosine, Covariance and RBF)?*

Question 5 *Is the NTD-based Cosine similarity better performing than the Barwise TF Cosine similarity in the structural segmentation task, and, when yes, in what features?*

The rationale of **Questions 4** and **5** is to study how the compression induced by NTD impacts the overall barwise similarity and homogeneity in the autosimilarities, and, more particularly, the contrast between similar and dissimilar bars (*i.e.* the difference in values between similar and dissimilar bars). Indeed, differences in performance between the similarity functions in Section 3.4 are explained by a higher contrast between similar and dissimilar bars in the Covariance and RBF autosimilarities compared to the Cosine autosimilarity. These differences in contrasts are empirically exhibited in Figure 4.9.

Question 6 *How KL- and IS-NTD impact segmentation results, compared to Euclidean-NTD, when the features exhibit energy discrepancies across frequencies, such as for STFT and Mel spectrograms?*

And how are KL- and IS-NTD performing when the energy discrepancies between frequencies are mitigated in the feature, such as energy-normalized chromagrams (CENS)?

Question 6 evaluates quantitatively the motivations behind the use of KL- and IS-divergences (*i.e.* aiming at reducing the impact of energy discrepancies between frequencies) in the structural segmentation task, via the NTD.

Chromagram, Euclidean-NTD

The first experiments with NTD [Mar+20] were obtained using chromagram as the feature representation, and the Euclidean-NTD. These results were computed with the CBM algorithm, applied to the Cosine autosimilarity matrix, and with the 4-bands kernel, which was shown to be poorly performing for segmentation in Chapter 3. Hence, Table 4.1 compares segmentation results obtained on the Cosine autosimilarity of chromagrams

with both 4-bands and 7-bands kernels, on RWC Pop. Factor W is fixed to I_{12}, as already discussed is Section 4.4.2.

Results in Table 4.1 indicate that NTD achieves better segmentation performance than the Barwise TF matrix, consistently with findings in [Mar+20], with both 4-bands and 7-bands kernels [9]. In addition, the 7-bands kernel obtains better performance than the 4-bands kernel for NTD, hence, for future experiments and as concluded in Chapter 3, we focus on the 7-bands kernel.

Results presented in Table 4.1 seem to answer **Question 5** positively, *i.e.* that the NTD-based Cosine similarity performs better for the task of structural segmentation than does the Barwise TF Cosine similarity.

Table 4.2 presents segmentation results with the different autosimilarities computed from the Q matrix. While differences in performance are important on the Barwise TF matrix, presented in Figure 3.12 of Chapter 3, the different autosimilarities obtain similar results here.

These results are a first answer to **Question 4**, indicating that NTD-based autosimilarities perform similarly with all similarity functions, with a small decrease for the Covariance similarity though.

Cosine	Kernel	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Barwise TF matrix	4-bands	43.88%	46.59%	44.71%	67.62%	71.01%	68.62%
	7-bands	47.00%	44.25%	45.16%	68.26%	63.98%	65.48%
Q of NTD	4-bands	54.13%	62.76%	57.71%	68.35%	79.90%	73.16%
	7-bands	**59.45%**	**64.76%**	**61.71%**	**73.50%**	**80.22%**	**76.36%**

Table 4.1 – Comparing results between Barwise TF-based and NTD-based Cosine autosimilarities, on the chromagram.

Autosimilarity	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Cosine	**59.45%**	64.76%	**61.71%**	**73.50%**	80.22%	**76.36%**
Covariance	54.51%	67.72%	59.53%	66.44%	83.00%	72.78%
RBF	58.30%	**65.37%**	61.24%	72.27%	**80.72%**	75.78%

Table 4.2 – Results when computing different autosimilarities on Q, on the chromagram.

9. Differences between results with the 4-bands kernel presented here and results presented in [Mar+20] can be explained by the use of different values for the dimensions, and by small corrections in the CBM since the publication of [Mar+20].

Chromagram, KL- and IS-NTD

As presented in 4.3.3, the motivation for the use of β-divergences is to better account for the dynamic of the signal's spectrum. In the chromagram used here though (see Section 2.2.4), normalization steps are applied to counteract the energy unbalance in the spectrogram. Thus, we expect less advantages on the use of β-divergences for chromagrams, which is confirmed in practice by the segmentation results presented in Table 4.3.

Hence, Euclidean distance seems more suited for this type of energy-normalized chromagram, which is a partial answer to **Question 6**.

Chromagram	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Euclidean-NTD	**59.45%**	**64.76%**	**61.71%**	**73.50%**	**80.22%**	**76.36%**
KL-NTD	54.99%	58.80%	56.56%	71.50%	76.27%	73.47%
IS-NTD	52.10%	56.27%	53.87%	70.59%	76.45%	73.11%

Table 4.3 – Segmentation results on the Cosine autosimilarity of Q with Euclidean-, KL- and IS-NTD, on the chromagram.

Mel and NNLM spectrograms, Euclidean-, KL and IS-NTD

If we turn towards Mel and NNLM spectrograms [Mar+22], it can be assumed that, without any normalization step, coefficients exhibit high variation in intensity which are not correlated to human perception. Nonetheless, the logarithm scaling in the NNLMS should dampen high-power values, and hence the NNLMS should be less sensible to large variations than the Mel spectrogram.

Segmentation results for both features and the different loss functions are presented in Table 4.4. These results show an advantage for the KL- and IS-NTD over Euclidean-NTD on both features, which may confirm that KL- and IS-divergences are better suited for Mel and NNLM spectrograms, partially answering to **Question 6**. In addition, both KL- and IS-NTD achieve better performance than the Barwise TF Cosine autosimilarity, which positively answers **Question 5**. This is consistent with findings in [Mar+22].

We point out the results obtained on the Mel spectrogram, where the Euclidean-NTD is the worst performing method, and notably worse than the results obtained on the Barwise TF matrix. It means that NTD-based Cosine autosimilarity is not always better than the Cosine autosimilarity of the Barwise TF, presenting a negative example to **Question 5**. Nonetheless, on the Mel spectrogram, KL- and IS-NTD obtain higher results than the

Barwise TF matrix, suggesting that NTD improves results when the model is suited to the characteristics of the input data.

Table 4.5 present results of KL-NTD with the different autosimilarities (results are equivalent for Euclidean- and IS-NTD): segmentation results obtained with the different autosimilarities are really close.

Still, as presented in Table 4.6, NTD-based autosimilarity is not always the best-performing case in terms of segmentation results. Indeed, results obtained with the RBF autosimilarity of the Barwise TF, with NNLM spectrogram, are higher in terms of $F_{0.5}$ than the best-performing NTD-based autosimilarity, obtained on the Cosine similarity of the KL-NTD.

In addition, results of the Barwise TF Log Mel spectrogram (hence, uncompressed representation of the song), are higher for both $F_{0.5}$ and F_3 than the best-performing NTD. The Log Mel spectrogram taking values in $\mathbb{R}$ (hence, potentially negative), we cannot compute NTD on this feature to compare.

Overall, the NTD is competitive with the State-of-the-Art [GS15b] in terms of F_3, and remains behind in terms of $F_{0.5}$. NTD outperforms the other unsupervised State-of-the-Art techniques segmentation-wise.

Feature	Kernel	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Mel	Barwise TF	**55.58%**	57.18%	55.88%	**74.17%**	76.61%	74.71%
	Euclidean-NTD	45.60%	48.08%	46.51%	68.30%	72.01%	69.66%
	KL-NTD	55.50%	60.13%	**57.42%**	73.69%	80.00%	**76.34%**
	IS-NTD	53.39%	**60.21%**	56.35%	71.17%	**80.78%**	75.32%
NNLMS	Barwise TF	50.89%	49.80%	49.92%	68.70%	66.98%	67.26%
	Euclidean-NTD	56.56%	57.75%	56.97%	74.05%	75.74%	74.63%
	KL-NTD	**60.34%**	63.84%	**61.79%**	**78.25%**	83.32%	**80.37%**
	IS-NTD	57.85%	**64.11%**	60.55%	75.33%	**83.43%**	78.83%

Table 4.4 – Segmentation results when computing Euclidean-, KL- and IS-NTD on Cosine similarity.

Importance of Dimension Selection

Finally, as an additional experiment, we present results where the dimensions F', S' and B' of NTD are not fixed or learned by cross-validation, but selected *a posteriori* as the dimensions maximizing $F_{0.5} + F_3$, *i.e.* a trade-off between the F measures at both tolerances. This condition, called "oracle condition" does not correspond to a realistic

Feature	Kernel	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Mel	Cosine	**55.50**%	60.13%	**57.42**%	**73.69**%	**80.00**%	**76.34**%
	Covariance	52.89%	**60.41**%	55.91%	68.81%	78.89%	72.90%
	RBF	53.75%	58.94%	55.94%	71.66%	78.50%	74.55%
NNLMS	Cosine	**60.34**%	63.84%	61.79%	**78.25**%	83.32%	**80.37**%
	Covariance	55.39%	**68.22**%	60.64%	67.80%	**84.36**%	74.53%
	RBF	60.17%	65.20%	**62.31**%	76.77%	83.54%	79.67%

Table 4.5 – Segmentation results for KL-NTD with the different autosimilarities.

Method	Conditions	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
NTD	KL-NTD, Cosine, NNLM	60.34%	63.84%	61.79%	78.25%	83.32%	80.37%
Barwise TF	RBF, NNLM	63.34%	66.33%	64.27%	76.53%	80.47%	77.80%
	RBF, Log Mel	64.32%	**70.01%**	66.52%	78.31%	**85.64%**	**81.16%**
Foote [Foo00]		42.03%	29.95%	34.48%	67.06%	47.66%	55.01%
CNMF [NJ13]		31.57%	28.11%	28.81%	50.68%	45.37%	46.53%
Spectral Clustering [ME14a]		49.21%	45.03%	45.01%	65.54%	60.56%	60.30%
Structural Features [Ser+14]		51.31%	38.02%	42.96%	74.40%	54.73%	62.15%
CNN [GS15b]		**80.36**%	62.68%	**69.70**%	**91.86**%	71.13%	79.34%

Table 4.6 – Comparison of the best-performing NTD (Cosine similarity of Q obtained with KL-NTD on the NNLM spectrogram) with the best-performing Barwise TF on the same feature (RBF on the NNLM spectrogram), the global best-performing Barwise TF (RBF on the Log Mel spectrogram), and the State-of-the-Art methods.

scenario, but presents a potential upper limit of NTD if some criterion was found to adequately dimension the parameters of NTD prior to the decomposition.

In this experiment, we focus on the Euclidean-NTD applied on chromagrams with $W = I_{12}$, because it is our least computationally expensive condition: firstly, one of the 4 factors is fixed, hence gaining one alternate in the optimization, and, secondly, chromagrams have a dimension equal to 12, which reduces complexity in the computation of update rules. In this experiment, dimensions S' and B', respectively number of columns of H and Q, take their values in $\{12, 16, 20, 24, 28, 32, 36, 40, 44, 48\}$, which is a finer grid (but still not an exhaustive search in all the parameters spaces).

Results, presented in Table 4.7, show a strong increase in performance for all the autosimilarity conditions. With these results, we conclude that providing a dimensioning

criterion for NTD would be a strong asset. Still, to this day, all our attempts in that direction were unsuccessful.

Dimension selection	Autosimilarities	$F_{0.5}$	F_3
Cross-validation	Cosine	61.71%	76.36%
	Covariance	59.53%	72.78%
	RBF	61.24%	75.78%
Oracle condition	Cosine	**74.25**%	**85.25**%
	Covariance	71.77%	79.60%
	RBF	73.91%	84.50%

Table 4.7 – Results of Euclidean-NTD with dimensions fitted by cross-validation (previous scenario) and in the oracle condition, on the chromagram.

Conclusions: NTD for Structural Segmentation

In this Section, the output of the NTD, and in particular the Q matrix, served as a barwise feature representation used in a segmentation context via the CBM algorithm. Experiments were conducted with the objective to answer 3 questions, for which we make the following conclusions.

Experimental conclusion 4 *The different autosimilarity strategies (Cosine, Covariance and RBF) used with NTD-based representations do not influence segmentation results as much as for the Barwise TF-based autosimilarities. In general, results are close, with a slight empirical advantage for the Cosine autosimilarity.*

Experimental conclusion 5 *Reported segmentation results show in every condition (with the exception of Mel spectrograms and Euclidean-NTD) an advantage of the NTD-based Cosine autosimilarity over the Barwise TF-based Cosine autosimilarity.*

We assume that these conclusions stem from a higher contrast between zones of high and low similarity in the Cosine autosimilarity of the NTD-based autosimilarity, compared to the Cosine autosimilarity of the Barwise TF.

This conclusion is empirically exhibited in Figure 4.9, where the NTD-based autosimilarities show more contrasted similarities. Nonetheless, in Figure 4.9, some NTD-based autosimilarities (in particular the RBF autosimilarity) exhibit stripe structures, which is closely related to the repetition criterion in structural segmentation, while the CBM algorithm focus on block structures, according to the homogeneity criterion.

In that sense, it should be informative to study the performance of NTD-based autosimilarities with a repetition-based algorithm (either by improving the CBM algorithm itself or by studying these autosimilarities with a different algorithm).

Experimental conclusion 6 *On our dataset, KL-NTD is the best-performing model for segmentation when studying Mel and NNLM spectrograms, and IS-NTD performs better than Euclidean-NTD.*

Euclidean-NTD is the best-performing method for the chromagrams, and this could be related to the fact that they are energy-normalized.

Considering the energy discrepancies in Mel and NNLM spectrograms result in higher segmentation scores, while using the KL and IS divergences is counter-productive when the feature already normalizes the energy.

Still, the optimal performance obtained with Barwise TF-based autosimilarities (*i.e.* the RBF autosimilarity), and with the NNLM and Log Mel spectrograms, are competitive with the optimal performance obtained with NTD (except in the unrealistic oracle condition). Nonetheless, in the light of these experimental conclusions, we conclude that NTD is an interesting paradigm for structural segmentation of music, which deserves to be further investigated and improved, as pointed out by the experiments in the oracle condition.

To further demonstrate the potential interest of NTD, and study more qualitatively the output of the decomposition, we study in Section 4.5.2 the other factors (W, H and $\mathcal{G}$) and, more particularly, their products, which form musical pattern, as a way to uncover patterns in music. This also allows us to qualitatively compare the different β values for audio signal processing with NTD, and test more extensively the Experimental conclusion 6, in different conditions.

4.5.2 NTD for Pattern Uncovering

As presented in Section 4.4.2, the product $P_{b'} = W\mathcal{G}_{::b'}H^\intercal$ for each $1 \leq b' \leq B'$ is analog to a barwise spectrogram, which we call "musical pattern". The previous section showed how the Q matrix, representing the song using the musical patterns as barwise features, can be used for structural segmentation.

This section focuses on the musical patterns themselves. Indeed, optimization-wise, each musical pattern represents some factorized content in the song, but this does not ensure that each pattern is musically relevant, or even interpretable. To this end, we

perceptually investigate the quality of patterns, by considering the NTD computed on STFT only. In particular, as NTD must be computed on nonnegative spectrograms, we consider NTD computed on the modulus of the STFT (magnitude spectrogram).

Until 2017, a task in the MIREX contest called "Discovery of Repeated Themes & Sections" was dedicated to pattern uncovering, but was restricted to the identification of patterns in symbolic format. In our scenario, patterns are spectrograms, and should be converted into MIDI representation, which is not straightforward and may alter the information.

In that sense, as in [SKG19], we turn towards the task of audio source separation to find evaluation metrics. Even if pattern uncovering is not the same task as source separation, it shares the idea of extracting relevant parts in an audio mixture and evaluating the quality of these extracted parts. Still, standard metrics are based on audio signals, while our patterns are real-valued spectrograms. In that sense, from a real-valued spectrogram $W\mathcal{G}_{::b'}H$, we first estimate the missing phase information, and then apply the Inverse STFT to result in an audio signal.

Reconstructing Phase Information

As presented in Section 2.2.4, STFT computes a complex-valued representation of signals. The Inverse STFT can be used to compute a signal from a complex-valued representation, but NTD is performed on real-valued spectrograms (in particular here, the magnitude spectrogram).

The argument of each complex STFT coefficient is the phase information, and must be estimated prior to signal reconstruction from the Inverse STFT. It is a common problem in signal processing [Bal10; Mag16] and in many engineering situations in general [CLS15; She+15; EHM16].

Phase Retrieval A first option, called "phase retrieval", consists of estimating the phase from the real-valued spectrogram only, but it is a difficult task, still unsolved. Phase retrieval has been largely studied for audio source separation, with the aim to estimate the quality of separated sources [Mag16; Via+21].

In the current context, we aim at identifying whether musical patterns are musically relevant or not, and interpretable in musical contexts. Thus, rather than presenting an exhaustive list of phase retrieval algorithms and seeking for the optimal phase retrieval algorithm for our specific context, we focus on the Griffin-Lim algorithm [GL84], still

largely used in practice.

This algorithm has the advantages of being conceptually simple, with theoretical guarantees regarding the improvement of the estimation at each iteration, and implemented in the *librosa* toolbox [McF+21]. Denoting as $Y = X.\Phi$ a complex-valued spectrogram of magnitude coefficients X and of phase information $\Phi = e^{i\phi}$, $\mathcal{F}$ and $\mathcal{F}^{-1}$ respectively the STFT and Inverse STFT operators, the idea of the algorithm is to iteratively 1) compute $\hat{Y}^{(t)} = (\mathcal{F} \circ \mathcal{F}^{-1})(Y^{(t)})$, *i.e.* the projection of the spectrogram into the audio domain, then re-projected in the time-frequency domain [10] and 2) compute the new estimate $Y^{(t+1)} = \frac{\hat{Y}^{(t)}}{|\hat{Y}^{(t)}|}.X$, *i.e.* the previous complex estimate with the original magnitude X.

The initial phase information Φ can be initialized at random or from a particular value. In this study, we only consider random initialization for the Griffin-Lim algorithm. Note here that any other phase retrieval algorithm could be used instead if musically-motivated, which could improve the final audio signals.

To sum up, for a pattern $P_{b'}$, we retrieve a complex-valued spectrogram $\tilde{P}^{GL}_{b'} = P_{b'}.\Phi_{b'}$ where $\Phi_{b'}$ is estimated using the Griffin-Lim algorithm.

Masking A second option is to use the phase information from the original spectrogram (of the song) as phase information for the pattern [SKG19]. Each musical pattern $P_{b'}$ acting as a factorization of the barwise real-valued spectrograms, the closer is $P_{b'}$ to a barwise spectrogram in the song, the closer is the phase information between this pattern and this bar.

In that sense, if a pattern indeed corresponds to a barwise factorization of the musical content in the original song, the phase information of a well-chosen bar could be used to transform the pattern in a complex-valued spectrogram. In practice though, it may not be as straightforward, as a pattern could also correspond to a part of a bar (such as a bass line among the entire barwise mixture).

Hence, as in the work of Smith *et al.* [SKG19], we instead turn towards source separation techniques, and, in particular, softmasking [Ben+03]. The idea behind masking is to apply a binary mask M on the original complex-valued spectrogram Y as the elementwise product $M.Y$. When a coefficient M_{ij} is equal to 1, the value of $(M.Y)_{ij}$ is unchanged. When M_{ij} equals 0, $(M.Y)_{ij} = 0$, discarding the original spectrogram content. Hence, only some particular complex time-frequency points of Y are kept in the masked spectrogram.

10. This operation makes sense because the STFT is not bijective, thus $\mathcal{F}^{-1}$ is not the mathematical inverse of $\mathcal{F}$.

Softmasking follows that idea, but M takes values in $[0, 1]$ instead, which brings more nuance than binary operations. In source separation, softmasking is applied to evaluate the impact of a particular source s_k among a set of sources, hence $1 \leq k \leq K$. Each source then defines a different mask M_k, where:

$$M_k = \frac{s_k}{\sum_{i=1}^{K} s_i}, \tag{4.28}$$

i.e. the impact of each source in the overall mixture. This softmask operator is optimal in source separation with respect to a least squares fit of the energy repartition between the different sources, but it may not be optimal in a perceptual sense [FJ12]. We do not explore further this question and restrict to the mask presented in Equation 4.28, while KL- and IS- divergences based masks are presented in [FJ12]. Least squares problems being based on the Euclidean distance, we conjecture that other masking techniques could better adapt to the characteristics of audio signals, and may also be better suited to compare the different loss functions for NTD.

Still, our context differs from source separation in the sense that we evaluate barwise patterns which may not be present in all bars. Hence, the first step in our scenario is to choose a bar of interest to evaluate this pattern. Some discussion about how to choose a bar can be found in [SKG19], but we restrict this discussion to a unique strategy, the "loudness" strategy in [SKG19], based on the maximal values of Q.

Recalling that matrix Q represents the relative importance of each musical pattern in the different bars, we choose, for pattern $P_{b'}$, the bar b such that $b = \arg\max_l Q_{lb'}$, *i.e.* the bar b with maximal presence of pattern $P_{b'}$ among all bars.

When such a bar is found, mask $M_{b'}$ is defined following Equation 4.29:

$$M_{b'} = \frac{Q_{bb'} W \mathcal{G}_{::b'} H^\intercal}{\sum_{i=1}^{B'} Q_{bi} W \mathcal{G}_{::i} H^\intercal}, \tag{4.29}$$

i.e. softmasking based on the presence of pattern $P_{b'}$ in bar b compared to the overall fit of bar b with NTD. Bar b is referred to as the **bar associated with pattern** $P_{b'}$.

This masking strategy is different from the one of Smith *et al.* [SKG19]: in their strategy, the mask is computed as: $M_{b'} = \frac{\mathcal{G}_{::b'} \times_1 W \times_2 H \times_3 Q_{:b'}}{\mathcal{G}_{::b'} \times_1 W \times_2 H \times_3 Q_{:b'} + \mathcal{X}}$, *i.e.* the contribution of pattern $P_{b'}$, at the song scale, compared to the original spectrogram of the song. The associated bar is selected in a post-processing stage.

Hence, in softmasking condition, a real-valued pattern $P_{b'}$ is used to compute a mask $M_{b'}$, itself used to compute a complex-valued spectrogram $\tilde{P}_{b'}^{mask}$ as $\tilde{P}_{b'}^{mask} = \mathcal{X}_{::b}.\Phi_b.M_{b'}$, where Φ_b represents the phase information of bar b in the original spectrogram.

Audio Evaluation Metrics

Once the phase is estimated for a pattern, a signal s can be obtained from the complex-valued spectrogram $\tilde{P}_{b'}$ by applying the Inverse STFT, $s_{P_{b'}} = \mathcal{F}^{-1}(\tilde{P}_{b'})$, using the *librosa* toolbox.

The objective is now to quantitively evaluate this audio signal by comparing it to an original barwise excerpt (or, in general, to quantitively compare two audio signals of same length). To this aim, several metrics were designed in the source separation context. We list in particular:

— SDR/SAR/SIR [VGF06], respectively defining the source-to-distortions, -artifacts and -interferences ratios, estimating objective perturbations of a signal compared to a clean version (or, in source separation settings, of the estimated sources with the ground-truth sources),

— TPS/IPS/APS [Emi+11], respectively Target-, Interference- and Artifacts-related Perceptual Score, which represent perceptual and subjective audio measures, designed to follow human perception. They are based on human evaluation in perceptual experiments.

For ease of use though, we only consider the SDR/SAR/SIR metrics [11], implemented in the *mir_eval* toolbox [Raf+14].

In source separation, estimated sources are compared with ground-truth sources, which are not available in our case. Instead, we can only make use of the original song as ground truth, and no obvious candidate stands out for the different patterns [12].

Hence, for each pattern $P_{b'}$, we compare its signal with the signal x_b of the associated bar b. In this pairwise comparison, and with the definitions of interferences (signal coming from other sources), noise (related to the sensors recording the signal) and artifacts (other

11. Still, it should be highly informative to evaluate our signals with TPS/IPS/APS metrics: the use of KL- and IS-divergences is motivated by their ability to better represent perceptually audio signals, by being less concentrated on high-intensity components. This kind of argument is also used to motivate perceptual metrics in [Emi+11], hence indicating that such metrics are better suited to compare the different loss functions for NTD, and in particular highlight the potential advantages of KL- and IS-divergences. This is left to future work, because we did not have the time to fully explore this lead.

12. The goal here is itself to evaluate if patterns are relevant.

perturbations of the signal, like distortion) in [VGF06], only artifacts and potentially noise are relevant in our case.

Hence, we focus on the SDR metric [13], which is a logarithmic version of the ratio between the squared-energy of the source and the sum of the squared-energies of the errors (interferences (= 0 here), artifacts and noise), as defined in Equation 4.30:

$$\text{SDR} = 10\log_{10}\frac{\|\text{energy of the source}\|^2}{\|\text{energy of artifacts} + \text{energy of noise}\|^2}. \tag{4.30}$$

To account for artifacts generated from the application of STFT and Inverse STFT, we apply the same transformation for x_b, *i.e.* starting from the original audio signal, x_b is obtained by applying $\mathcal{F}^{-1} \circ \mathcal{F}$.

We acknowledge that this strategy does not allow to evaluate the pattern in itself, but rather its similarity to some original content in the song. If the pattern corresponds to a small part of the original bar (for instance, a bass line), SDR may be low. Still, a high SDR indicates that the pattern is indeed musical and interpretable. In addition, this strategy allows us to compare Euclidean-, KL- and IS-NTD. Due to the lack of other pattern uncovering methods, we are not be able to compare these results with State-of-the-Art ones.

Patternwise Experiments

Finally, we run NTD with $F' = 32, S' = 12, B' = 10$ on the STFT of the song *Come Together* by The Beatles [14]. This results in 10 patterns for Euclidean-, KL- and IS-NTD. The SDR scores are then computed on each pattern $P_{b'}$, compared to its associated bar. For each pattern, the complex-valued spectrogram $\tilde{P}_{b'}$ is computed either with the Griffin-Lim algorithm or with softmasking, resulting in 2 conditions.

Results are presented in Table 4.8, and show a clear advantage of softmasking over Griffin-Lim. Still, the Griffin-Lim algorithm is a blind pattern estimation technique, and hence it retrieves phase information associated with the pattern itself, and not retrieved from the original song (which may artificially add information). Comparing the different NTDs, KL slightly outperforms both other loss functions, both with Griffin-Lim and

13. We could eventually consider SAR, depending on what is considered to be sensor noise. In practice, on early experiments, both SDR and SAR were equal, suggesting that the noise term is zero here.

14. Dimensions were chosen empirically, but are equal to the ones used in Figure 4.8, except for F', the frequency-related one.

softmasking. Euclidean- and IS-NTD obtain close results with both phase retrieving techniques.

While SDR is informative, conclusions should be taken with care: SDR is obtained as an energy ratio, and softmasking is computed via an optimal least squares fit, which are both Euclidean distance-based. Hence, these metrics do not account for intensity discrepancies between frequencies, which was our motivation for the use of KL and IS divergences in the first place. Still, even in this scenario, KL-NTD outperforms Euclidean-NTD with both Griffin-Lim and softmasking, which goes in the same direction as **Experimental conclusion 6**.

Phase estimation	NTD	SDR
Griffin-Lim	Euclidean-NTD	-20.81 ± 2.44
	KL-NTD	**-17.69** ± 3.03
	IS-NTD	-20.47 ± 3.48
Masking	Euclidean-NTD	16.71 ± 5.10
	KL-NTD	**25.94** ± 5.48
	IS-NTD	19.02 ± 8.71

Table 4.8 – Average and standard-deviation of the SDR scores for the 10 patterns obtained from NTD on the STFT of the song *Come Together*. Note that two Euclidean-NTD patterns resulted in a mask with every entry equal to one, and these patterns were discarded in the computation of SDR (the spectrogram of the pattern is exactly the spectrogram of the bar in this scenario, which is not informative).

Evaluating the Entire Song

NTD is based on the optimization of the decomposition of the song as a whole, hence it can be used to study the entire song. In this experiment, we compute $\hat{\mathcal{X}} = \mathcal{G} \times_1 W \times_2 H \times_3 Q$, which corresponds to the real-valued spectrogram of the factorization at the song scale, and transform it into an audio signal.

To account for phase information, we use the Griffin-Lim algorithm as previously presented, but not softmasking, which seems unsuited at the song scale (what should constitute the denominator of the mask?).

Instead, we directly use the phase information of the original song Φ, and reconstruct the signal from the complex-valued spectrogram $\hat{\mathcal{X}}.\Phi$. This is a new condition, which in our opinion is more relevant here, as $\hat{\mathcal{X}}$ is assumed to reconstruct adequately the original magnitude spectrogram $\mathcal{X}$. SDR scores for these two audio signals, at the song scale, are

presented in Table 4.9.

Results at the song scale seem to conclude in favor of KL-NTD, for both phase retrieval techniques. Euclidean-NTD and IS-NTD result in contradictory outcomes depending on the phase retrieval technique.

As for previous conclusions, it is hard to conclude firmly, but these results give some hints: even if the metric seems tailored for Euclidean distance-based techniques, KL-NTD outperforms Euclidean-NTD, going in the direction of **Experimental conclusion 6**.

In addition, it is pretty clear in the light of the results that the Griffin-Lim algorithm is introducing numerous artifacts and noise, and that using the original phase should be preferred to estimating it from scratch. The original phase could also be used as initialization for the Griffin-Lim algorithm, which is not tested here.

In addition, results at the song scale are lower than at the pattern (*i.e.* bar) scale, which may indicate that the global mixture of patterns is not sufficient for some bars. This is expected though, as each estimated pattern is compared to its closest bar.

Phase estimation	NTD	SDR
Griffin-Lim	Euclidean-NTD	-38.47
	KL-NTD	**-34.53**
	IS-NTD	-36.99
Original Phase	Euclidean-NTD	4.35
	KL-NTD	**6.08**
	IS-NTD	2.51

Table 4.9 – SDR when reconstructing the signal of the entire song from the whole NTD, for *Come Together*.

Listening to Audio samples

This entire subsection being focused on evaluation of audio signals, a part of the study consists of listening to the factorization results. Audio examples cannot be presented in the manuscript, but some experiments are made available online at the following link: https://ax-le.github.io/resources/examples/ListeningNTD.html.

Perceptually, with Griffin-Lim, Euclidean-NTD results in less relevant outputs than KL- and IS-NTD, while it is hard to differentiate the techniques with softmasking. This is somewhat consistent with previous SDR scores and the **Experimental conclusion 6**, with KL-NTD being the most perceptually relevant technique musically speaking.

Perceptually, it seems that Euclidean-NTD focuses on low-frequencies, while IS-NTD catches mostly high-frequencies. In that sense, it is hard to conclude in favor of one of these methods in particular. Finally, these results are encouraging towards future research in pattern enhancement and study.

Conclusions: NTD for Pattern Uncovering

In conclusion, studying the uncovered patterns confirms that the NTD results in a part-based and interpretable representation of the original musical content.

In light of these experiments, and following the **Experimental conclusion 6**, it seems that KL-NTD is the most suited optimization paradigm here, resulting in a good balance between low- and high-frequency components, while Euclidean- and IS-NTD seem to respectively focus on low- and high-frequency components.

However, these conclusions are to be taken carefully. When using softmasking, Euclidean- and KL-NTD are close in performance. Future experiments with different masking conditions and different metrics (such as TPS/IPS/APS [Emi+11]) would be useful to confirm or contradict these results.

Finally, further experiments should be designed to study to what extent patterns represent exactly the content of particular bars/sections, or, conversely, represent only particular components of some bars (such as focusing on an instrument).

4.6 Conclusions

This chapter has presented a multilinear dimensionality reduction technique called Nonnegative Tucker Decomposition, performed on barwise representation of the original spectrogram of music pieces. By introducing this technique mathematically and its interpretation on music, we have demonstrated a potential relevance of this model for music analysis.

Experimental results on the task of structural segmentation and on a pattern uncovering framework confirmed in practice this theoretical interest. Finally, we have also introduced two NTD algorithms, computing the decomposition with respect to both Euclidean distance and β-divergences.

Experimental results presented in this chapter are based on unsupervised NTD, while NMF, more studied in MIR and relatively similar to NTD, has shown great improvements with constraints (such as sparsity [OP14], constraints on the shape of factors [VBB09;

Che+16]), semi-supervision/dictionary learning [Ewe+16; HWL21; WMC22], and coupled factorizations [Mes+15], *i.e.* factorizations where at least one factor is shared between several (otherwise independent) factorizations.

In particular, depending on the factor considered, these techniques could be combined, for instance in trying to learn or share factors W and H on several songs, acting as high-scale dictionaries, while adapting $\mathcal{G}$ and Q on each song, for instance with sparsity or prior constraints.

NMF algorithms were also designed to fit several loss functions in the same optimization paradigm [GLT21], which could be of great interest for future NTD algorithms, to mitigate between the different loss functions.

In that sense, we conclude that NTD seems to be a promising method for novel MIR techniques, and notably improving performance in the structural segmentation task, which are already close to those of the State-of-the-Art [GS15b] for the 3s tolerance. In a context of source separation, NTD could additionally be used, for instance recasting the Multichannel NMF [OF09] model for an instantaneous mixture of sources.

Chapter 5

Linear and Nonlinear Barwise Compression Schemes

Synopsis

This chapter presents new barwise compression schemes, used in place of the NTD. These compression schemes are standard linear compression schemes and nonlinear AutoEncoders. Compressed representations at the barscale then serves for structural segmentation.

5.1 Introduction

Autosimilarity matrices and the CBM algorithm were introduced in Chapter 3 as tools for segmenting a song. Autosimilarity matrices, originally computed from the raw barwise feature representation of the song, were extended in Chapter 4 to the use of NTD, a multilinear dimensionality reduction technique. NTD is used with the objective to compute barwise compressed representation of the song.

The current chapter introduces different barwise compression methods, which may be used instead of NTD for computing barwise compressed representation of the song. We divide these compression techniques in two sections:

— The first section presents linear compression methods, namely NMF and Principal Component Analysis (PCA), also referred to as low-rank factorization methods. These methods are standards for data analysis [Jol02; Mar12; Gil20], and are generally able to uncover motifs and redundancy in data in an unsupervised fashion. Thus, studied for barwise compression of music, these methods serve so as to deepen the conclusions obtained previously with NTD by relaxing the constraints (non-negativity and NTD structure), and extend the barwise compression framework to standard linear methods.

— The second section presents nonlinear compression methods, with neural networks. In particular, this chapter presents the "Single-Song AutoEncoding" paradigm, introduced in [MCB22c], which consists of AutoEncoders optimized on single songs instead of large datasets.

  Thus, in this paradigm, neural networks are used as blind compression methods at the level of the song, allowing to benefit from the large neural network literature and most recent developments. Finally, Single-Song AutoEncoding extends and studies in more depth the unsupervised barwise compression paradigm, and, hopefully, may improve segmentation results.

Hence, this chapter aims at further investigating the ability of compression schemes to disambiguate the raw feature-wise barwise similarity, and presents segmentation results which outperform those previously obtained with the NTD. The contributions reported in this section are twofold:

— Methodological: while NMF and PCA are standard compression methods, they were, to the best of our knowledge, never used at the barscale, studying barwise redundancy. In addition, while AutoEncoders are also standard methods to learn

general representations, we introduce their use as song-dependent compression. Indeed, rather than trying to learn a general representation for similarity computation, AutoEncoders are employed here at the song scale, only for compression abilities.

— Experimental: as previously introduced for NTD, barwise compressed representations are used to compute autosimilarity matrices, themselves used for structural segmentation via the CBM algorithm. The best results reported here outperform the previous NTD segmentation results.

This section resulted in one publication [MCB22c], and both the code and the conducted experiments are included in the open-source *BarMusComp* toolbox [MCB22b].

5.2 Barwise TF matrix (reminder)

The Barwise TF representation was introduced in Section 2.4.2, but, as it is of particular importance in this Section, and for clarity, let us reintroduce again this representation here.

The Barwise TF representation aims at representing the spectrogram as barwise vectors of Time-Frequency components. Time and frequency only represent one dimension in this matrix. In the tensor point of view, the Barwise TF matrix can be seen as the unfolding of the TFB tensor (introduced in Chapter 4) along the bar mode, which is the last mode. The Barwise TF matrix is represented in Figure 5.1.

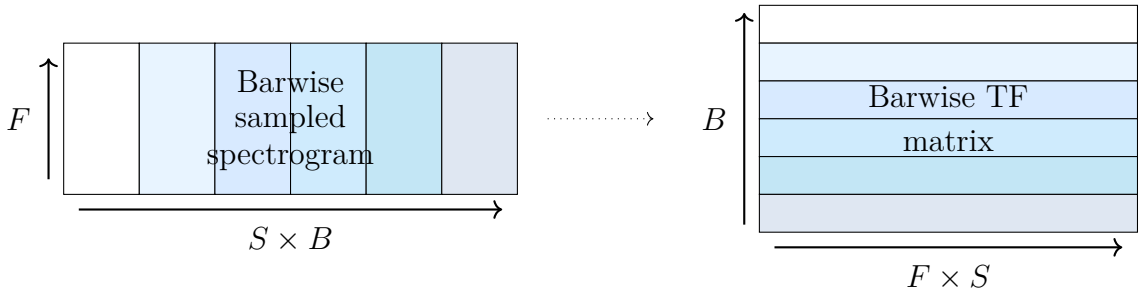


Figure 5.1 – Barwise TF matrix.

In this Section, the Barwise TF matrix is denoted as $X \in \mathbb{R}^{B \times FS}$, with F the size of the frequency dimension, S the number of frames at the inner-bar scale (S being the "subdivision" parameter, controlling he number of frames in each bar), and B the number of bars. In particular, X can be nonnegative, for chromagrams, Mel and NNLM

spectrograms, or possibly negative, for instance with Log Mel and MFCC spectrograms.

5.3 Low-Rank Factorizations

The first studied class of compression methods are low-rank matrix factorization methods [Mar12]. Low-rank factorizations methods compute an approximation of the original matrix X with lower-rank matrices.

In particular, we compare two standard models for linear dimensionality reduction: NMF, consisting of representing a matrix with two nonnegative matrices, and Principal Component Analysis (PCA), consisting of finding the low-rank approximation which maximally scatters the original data points in the projection. The main difference between both models is the nonnegativity constraint of NMF, which is absent in the PCA framework.

5.3.1 Barwise NMF

Nonnegative Matrix Factorization, or NMF, was already introduced in Section 4.2.3: given a nonnegative matrix M, NMF computes two nonnegative matrices U and V such that $M \approx UV$.

> In Barwise NMF, the idea is to approximate the Barwise TF matrix $X \in \mathbb{R}_+^{B \times FS}$ as the product of two matrices $Q \in \mathbb{R}_+^{B \times B'}$ and $P \in \mathbb{R}_+^{B' \times FS}$ with B' the dimension of compression (parameter of the decomposition).

Barwise NMF is computed via the following optimization problem, subject to loss function $d()$:

$$\underset{Q,P \geq 0}{\arg\min}\ d(X, QP). \tag{5.1}$$

This setting is particularly close to the NTD paradigm. In fact, as presented in Section 4.3, algorithms for NTD alternate between factors by fixing all of them but one.

The particular subproblem for the barwise mode of the tensor is $\underset{Q}{\arg\min}\ d(\mathcal{X}_{(3)}, Q\mathcal{G}_{(3)}(W \otimes H)^\intercal)$. When setting $P := \mathcal{G}_{(3)}(W \otimes H)^\intercal$, the optimization subproblem with respect to Q is exactly the same between the NTD presented before and the Barwise NMF introduced here [1].

1. Which is actually the rationale of the Q notation in the Barwise NMF setting.

Still, while the P factor is composed of three factors in NTD, it is represented by a unique matrix in Barwise NMF. Hence, contrary to NTD, Barwise NMF does not assume structure in the time-frequency dimension, which presents both advantages and disadvantages.

The main disadvantages are that, by losing the structure, Barwise NMF is less constrained on the time and frequency modes, which may result in less relevant and interpretable decomposition. In addition, it is not possible here to extend the model by imposing specific constraints and/or dictionary learning on the different factors, which seems to be a promising future direction for NTD.

The main advantages are that Barwise NMF can directly profit from the large NMF literature, that NMF depends on only one dimension parameter B' instead of three, correlated with the number of barwise patterns, and that NMF is faster than NTD. One could imagine using Barwise NMF for the initialization of NTD, or in strategies to fix the barwise dimension B' prior to the decomposition in NTD. In practice, dimensioning NMF is a hard problem, generally solved by fixing the dimension B' empirically or thanks to prior knowledge [Gil20].

Barwise NMF is computed in an alternating scheme, using the algorithms presented in Section 4.3, *i.e.* HALS for the optimization of the Euclidean-NMF problem: $\underset{Q,P\geq 0}{\arg\min} \|X - QP\|_2^2$ and MU for the β-divergence-NMF: $\underset{Q,P\geq 0}{\arg\min}\, d_\beta(X, QP)$, defining in particular KL- and IS-NMF when $\beta = 1, 0$ respectively.

Similarly to NTD, Q can be studied for structural segmentation, and P for pattern uncovering. Still, we restrict this study here to structural segmentation, to streamline the conclusions.

5.3.2 Principal Component Analysis - PCA

PCA is probably one of the most standard linear dimensionality reduction method. Originally introduced in 1901 by Pearson [Pea01], and reintroduced under the name of PCA by Hotelling [Hot33], PCA has been studied in a lot of applications, see [Jol02] for a comprehensive presentation of the technique and its use.

PCA was already used for structural segmentation in [TEF13], but not in a barwise context. PCA is related to the Singular Value Decomposition (SVD), whose origins trace back to the 19$^{\text{th}}$ century [Ste93], which is also a standard matrix decomposition tool.

Denoting as $\bar{x}$ the average of all bars in the song, the rationale of PCA is to project the centered data matrix $X^c = X - \mathbb{1}_B \bar{x}^\intercal$ on an orthonormal basis such that each dimension of this subspace maximizes the variance of the projected data, hence the idea of "principal components".

The principal components are ordered by the variance of the projection. Mathematically, it means finding vectors $(p_1, p_2, ..., p_n)$ such that $p_1 = \underset{\|p\|_2=1}{\arg\max} \sum_i (X_i^c p)^2 = \underset{\|p\|_2=1}{\arg\max} \|X^c p\|_2$, $p_2 = \underset{\|p\|_2=1}{\arg\max} \|(X^c - X^c p_1 p_1^\intercal)p\|_2$, etc. It is proven that such vectors correspond to the eigenvectors of $X^{c\intercal} X^c$ [And58].

PCA is generally used as a low-rank approximation technique, by projecting the data on the r first principal components. Denoting as: $X^{c\intercal} X^c = D \Lambda D^{-1}$ the eigenvalue decomposition of $X^{c\intercal} X^c$, the projection corresponds to: $X^c D_{1:r}$. PCA may be computed using the SVD of X^c, as implemented in the *scikit-learn* toolbox [Ped+11], used in this thesis.

In our context, PCA aims at computing barwise compressed representation, hence projecting X on the subspace generated by the first B' principal components, reducing the dimensionality of the time-frequency mode. Thus, a matrix $Q \in \mathbb{R}^{B \times B'}$ is computed as $Q = X^c D_{1:B'}$, which can be used for structural segmentation of music.

The D matrix can be possibly negative. Hence, we do not expect these vectors to be interpretable or to consist of part-based representation of the song, thus they are not studied in the pattern uncovering context.

By definition, Q is a centered matrix. Hence, the Cosine and Covariance autosimilarities of Q are equivalent, and therefore we restrict our study to Cosine and RBF autosimilarities.

5.3.3 Structural Segmentation Experiments

Both NMF and PCA compute a barwise compressed representation $Q \in \mathbb{R}^{B \times B'}$. This representation can now be used in the context of structural segmentation, via the CBM algorithm, and examples of Cosine autosimilarities are presented in Figure 5.2.

Still, one has to dimension the parameter B' prior to the decomposition. It is well-known that selecting the number of components for NMF is a difficult problem, generally leading to manual tuning or dedicated heuristics [Gil20; Neb+21]. Note that this was also a problem with NTD. In that sense, we compare values for $B' \in \{8, 16, 24, 32, 40\}$, as for

NTD.

Even if dimension selection heuristics are standard for PCA (such as the elbow method), as no obvious candidate heuristic stood out relatively to the quality of segmentation, and for fair comparison with the other techniques (NTD and NMF), PCA is tested with the same B' values. A clever dimension selection method could be studied in future work.

As detailed in Appendix A.2, the dimension is chosen via two-fold cross-validation for the RWC Pop dataset (odd and even songs) and by learning/testing on two subsets for SALAMI.

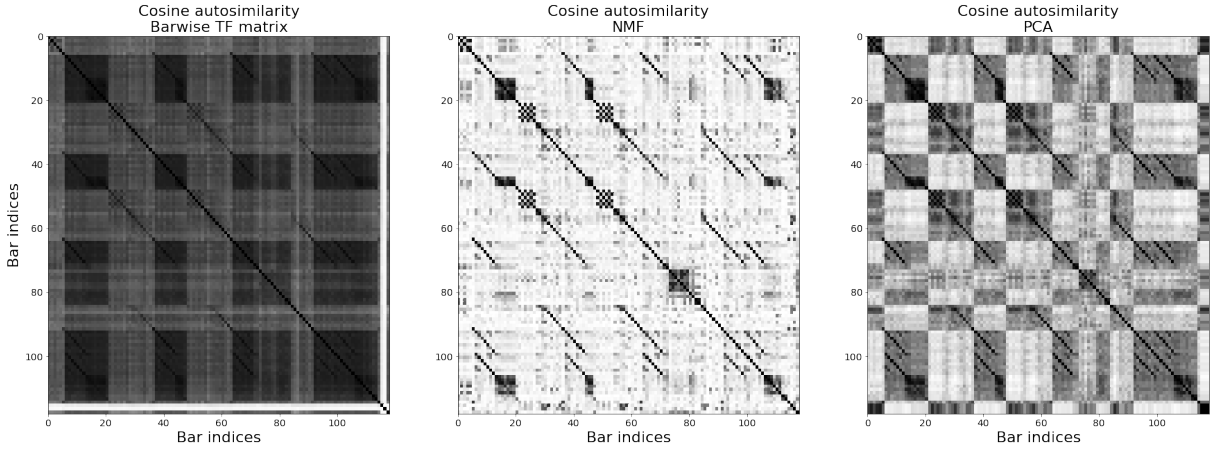


Figure 5.2 – Examples of Cosine autosimilarities of both the NMF and PCA, compared with the raw Barwise TF Cosine autosimilarity. These autosimilarities are computed on the NNLMS of the song *Pop01* of RWC Pop, with $B' = 24$.

NMF is initialized using the Nonnegative Double Singular Value Decomposition (NNDSVD) routine [BG08]. This routine performs an SVD on X, keeps the nonnegative part of the B' first left and right singular vectors of the SVD, and computes a second SVD on the products of these B' singular vectors. This routine is deterministic, and ensures a good nonnegative reconstruction of the original data X. The initialization largely impacts the output of NMF [BG08; Gil20], mostly due to the fact that perfectly solving NMF is NP-Hard [Vav10], and that most algorithms only guarantee convergence towards stationary points.

NMF is computed via the *nn_fac* toolbox [MC20], developed during this thesis, and PCA via the *scikit-learn* toolbox [Ped+11]. Both PCA and NMF are relatively fast algorithms, compared to NTD for instance. As an example, on an Intel® Core(TM) i7 CPU, decomposing the song *POP01* with $B' = 16$ for the Nonnegative Log Mel (NNLM)

spectrogram takes approximately 3 seconds for the Euclidean-NMF, 6 seconds for the KL-NMF and 13 seconds for the IS-NMF, while it takes less than a second for PCA.

We formulate some working questions, very similar than the ones formulated for NTD, which are tested for the two methods (NMF and PCA), on the relevant features (chromagrams, Mel and NNLM spectrograms for NMF, due to the nonnegativity, and additionally MFCC and Log Mel spectrograms for PCA), and with Euclidean-, KL- and IS-NMF.

Question 7 *With the current version of the CBM algorithm applied to NMF- and PCA-based autosimilarities, how are segmentation performance impacted by the similarity functions (Cosine, Covariance and RBF)?*

Question 8 *Are the NMF- and PCA-based Cosine similarity better performing than the Barwise TF Cosine similarity in the structural segmentation task, for all features? And compared to NTD?*

Question 9 *How KL- and IS-NMF impact segmentation results, compared to Euclidean-NMF, when the feature exhibits energy discrepancies between frequencies, such as for STFT and Mel spectrograms?*

And how are KL- and IS-NMF performing when the energy discrepancies between frequencies are mitigated in the feature, such as energy-normalized chromagrams?

Additionally, are these results consistent with the one obtained for NTD-based autosimilarities?

RWC Pop Dataset

Starting with PCA, **Question 7** is studied through the different autosimilarities, on the RWC Pop dataset. We recall that the PCA is centered, hence the Cosine and Covariance autosimilarities are equivalent. **Question 7** is then restricted to the Cosine and RBF autosimilarities.

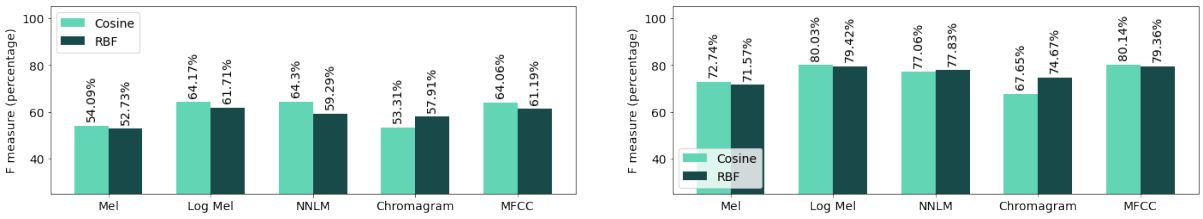
Results are presented in Figure 5.3, and, except for chromagram, the Cosine autosimilarity is the best-performing model for every feature and both tolerances. We do not explain the difference in trend for the chromagram. Still, we conclude with relative confidence here that the Cosine similarity is the best-performing autosimilarity for PCA, as the chromagram, even with the RBF autosimilarity, obtains among the lowest performance.

The results are more ambiguous for NMF, as presented in Figure 5.4: the RBF autosimilarity is the best autosimilarity in 2 among 3 features (NNLMS and chromagrams),

and Cosine is the best for Mel spectrograms. Still, in particular for the F_3 metric, results are almost similar between Cosine and RBF.

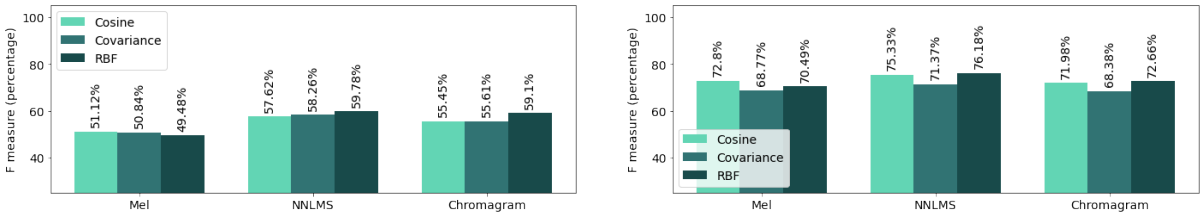
Trying to solve this ambiguity, we study results for KL- and IS-NMF, but we only present results for the KL-NMF in Figure 5.5 (results for the IS-NMF are similar). Results in this Figure are either similar between the autosimilarities (as for the PCP and the Mel spectrogram) or largely in favor of Cosine similarity (as for NNLM).

Hence, for both NMF- and PCA-based autosimilarities, the Cosine autosimilarity seems to be a good similarity choice. We then decide to restrict further experiments with this similarity function in Figure 5.6, for the different features (Mel, Log Mel, NNLM, Chromagram and MFCC), trying to answer to **Question 8**, and for Euclidean-, KL- and IS-NMF, studying **Question 9**.



(a) $F_{0.5}$. (b) F_3.

Figure 5.3 – Segmentation results for PCA, on the different autosimilarities.

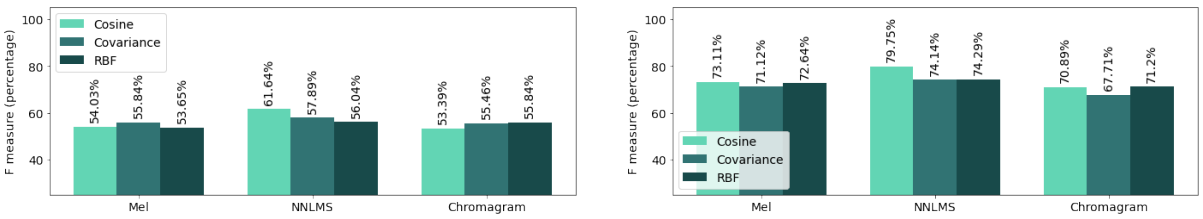


(a) $F_{0.5}$. (b) F_3.

Figure 5.4 – Segmentation results for Euclidean-NMF, on the different autosimilarities.

In Figure 5.6, except for the Mel Spectrogram, better performance are achieved with compressed representations, which motivates the use of compressed representations. We recall that, in results presented in Section 4.5.1, Euclidean-NTD was less performing than the Barwise TF, but KL- and IS-NTD obtained better performance than the Barwise TF. A similar behavior may appear for NMF.

Results for Euclidean-, KL- and IS-NMF are presented in Figure 5.7, in comparison



(a) $F_{0.5}$. (b) F_3.

Figure 5.5 – Segmentation results for KL-NMF, on the different autosimilarities.

with the results of the Barwise TF and of the best NTD [2]. KL-NMF seems to be the best-performing model for the NNLMS, IS-NMF is the best for Mel, with close results to KL-NMF, and Euclidean-NMF seems to be the best for chromagrams. Once again, adapting the loss function to the characteristics of the representation seems to lead to better segmentation results.

Comparing NMF and NTD, conclusions depend on the feature. For the NNLMS, results may be considered similar for both NTD and NMF, while, for the Mel spectrogram, opposite trends appear depending on the loss function (Euclidean-NMF outperforms Euclidean-NTD, but KL-NTD outperforms KL-NMF). NTD outperforms NMF for chromagrams, which may be explained by the fact that, for NTD, W was fixed to the identity matrix, hence reducing the parameter space.

Overall, the best performance for $F_{0.5}$ are obtained with the PCA, on 3 features: the Log Mel, the NNLM and the MFCC. Both of these representations, the Log Mel and the MFCC, can be negative, which make the comparison with NTD and NMF irrelevant.

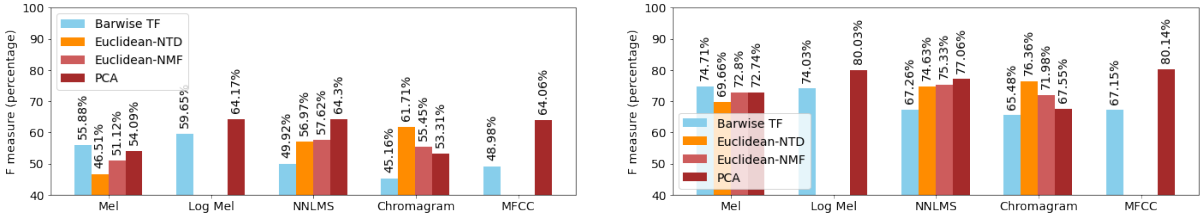
For the NNLMS, opposite conclusions can be drawn depending on the metric: the advantage of PCA over NMF at $F_{0.5}$ (respectively 64.30% and 61.64%) is inverted at F_3 (respectively 77.06% and 79.75%). Still, as a general trend, PCA obtains better performance.

The all-time best-performing technique remains the RBF autosimilarity of the Log Mel spectrogram, as presented in Table 5.1, but, at the same time, the compressed representations obtain in almost every condition better results with the Cosine autosimilarity.

Hence, while the aforementioned compression methods are not able to improve the best level of performance of the CBM algorithm, the results are less dependent on the choice of the similarity function, which probably indicates better notions of homogeneity and of contrast in the Cosine autosimilarity, *i.e.* representations which are more suited to

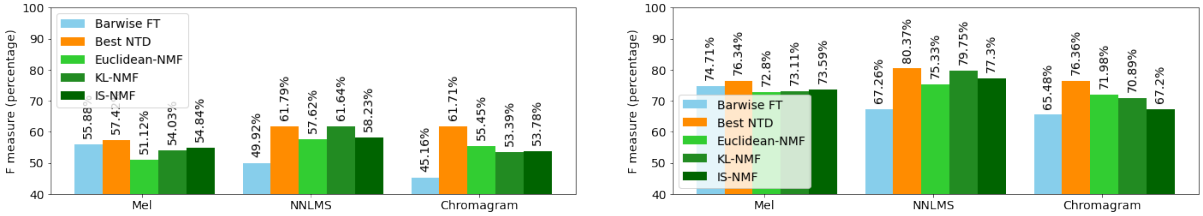
2. Euclidean-NTD for chromagrams and KL-NTD for Mel and NNLM spectrograms.

represent the structure in the song as so.



(a) $F_{0.5}$. (b) F_3.

Figure 5.6 – Segmentation results with the Cosine similarity, computed on the Q matrix obtained from PCA and Euclidean-NMF, compared with the segmentation results on the raw Barwise TF and with Euclidean-NTD.



(a) $F_{0.5}$. (b) F_3.

Figure 5.7 – Results for NMF with different loss functions, on the Cosine similarity.

Method	Best conditions	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Barwise TF	Log Mel, RBF	**64.32%**	**70.00%**	**66.52%**	78.31%	**85.64%**	**81.16%**
NTD	NNLM, KL	60.34%	63.84%	61.79%	78.25%	83.32%	80.37%
NMF	NNLM, KL	61.21%	62.99%	61.64%	**79.02%**	81.61%	79.75%
PCA	Log Mel	61.86%	67.78%	64.17%	76.96%	84.73%	80.03%

Table 5.1 – Comparison of the best-performing linear compression methods.

SALAMI Dataset

To broaden the previous experimental conclusions on another dataset, we consider experiments on the SALAMI dataset. To focus conclusions and due to the size of this dataset, we only considered the NNLM spectrogram for NMF (with the three loss functions), and both NNLM and Log Mel spectrograms for PCA (the latter being was the best of condition for PCA). Focusing on the Cosine autosimilarity, results are presented in Table 5.2.

An important conclusion on the SALAMI dataset is that performance are closer between the different conditions compared to those obtained on the RWC Pop dataset. Of particular interest, while the compression algorithms do not increase F measures as much on the SALAMI dataset compared to the raw Barwise TF-based results, compression alters the precision/recall trade-off.

Indeed, the precision and the recall are balanced for the Barwise TF-based results, while the precision is lower than the recall for compressed-based autosimilarities, suggesting over-segmentation. These differences stem from a higher precision with compressed representations (recall values are similar), hence suggesting that additional boundaries are obtained.

This result may indicate a higher contrast in the compression-based autosimilarity matrix: the CBM algorithm may increase the number of boundaries estimated if the block structure is more obvious in the autosimilarity, *i.e.* if the differences between similar and dissimilar bars are higher, leading to higher novelty between homogeneous sections. Still, this shift in trade-offs was not observed on the RWC Pop dataset.

Overall, compressed representation result in higher F-measures than Barwise TF-based results.

Feature	Compression method		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Log Mel	Barwise TF*		35.89%	34.64%	34.17%	**53.61%**	51.62%	51.03%
	PCA		**37.06%**	**56.74%**	**43.55%**	52.35%	**81.07%**	**61.79%**
NNLM	Barwise TF*		38.86%	41.63%	38.93%	**58.45%**	62.68%	58.63%
	PCA		36.12%	**57.75%**	43.06%	50.54%	**81.57%**	60.45%
	NMF	Euclidean	38.49%	51.05%	42.80%	55.82%	74.30 %	**62.15%**
		KL	**39.01%**	52.11%	**43.42%**	55.41%	74.53%	61.84%
		IS	37.59%	52.25%	42.66%	54.38%	75.98%	61.84%

Table 5.2 – Segmentation results of NMF and PCA on SALAMI, with the Cosine autosimilarity. *We recall that the Barwise TF condition means that there is no compression.

Conclusions for Barwise NMF and PCA

Experimental conclusion 7 *The different similarity functions (Cosine, Covariance and RBF) do not influence segmentation results of PCA- and NMF-based autosimilarities as much as for the Barwise TF-based autosimilarities. In general, results are similar, with an overall empirical advantage for the Cosine autosimilarity.*

This trend follows the experimental conclusions obtained with NTD, and suggests that compressed representations do not require further transformations to exhibit the structure.

Experimental conclusion 8 *Reported segmentation results show in every condition (with the exception of Euclidean-NMF applied on Mel spectrograms for RWC Pop) an advantage of the NMF- and PCA-based Cosine autosimilarities over the Barwise TF-based Cosine autosimilarity.*

Compared to NTD, PCA obtains similar or higher performance, while NMF obtains lower or similar performance.

As for NTD, we assume that these conclusions stem from a high contrast between zones of high and low similarity in the Cosine autosimilarities, disambiguating the boundaries (according to the homogeneity criterion) compared to feature-wise similarity.

Experimental conclusion 9 *On RWC Pop, considering NMF only, KL- and IS-NMF obtain better results than the Euclidean-NMF on both Mel and NNLM spectrograms, and the Euclidean-NMF is the best method for the chromagrams, which, we recall, are energy-normalized. This trend follows the trend of NTD-based autosimilarities, with the different loss functions.*

Conversely, all methods perform similarly on the SALAMI dataset.

Once again, adapting the loss function to the characteristics of audio signals leads to better segmentation results on the RWC Pop dataset.

5.4 Single-Song AutoEncoders

AutoEncoders (AE) are neural networks, which, by design, perform unsupervised dimensionality reduction. Throughout the years, AE have received increasing interest, notably due to their ability to extract relevant latent representations without the need for large amount of annotations.

Recently, AEs also showed great results as generation tools [Eng+17; Roc+18]. Still, as presented in [Roc+18], PCA and AE are competitive as compression schemes, and PCA even leads to lower reconstruction error when AE are too "simple" (in particular when networks are linear or "shallow", *i.e.* with only a few layers).

In our context, AEs may be used for barwise compression, in place of the previous NTD, NMF and PCA schemes. Therefore, AEs are developed in the context of unsupervised compression schemes at the song scale, which defines the "Single-Song AutoEncoding" paradigm. This paradigm is novel to the extent that AEs are generally used so as to learn a general latent representation on a dataset.

This section is dedicated to presentation and study of Single-Song AutoEncoding. In a first part, this section introduces the motivations and formalism of this paradigm. Then, two sets of experiments are conducted: the first studies Single-Song AutoEncoders for structural segmentation, and the second introduces some extensions of the paradigm with examples.

5.4.1 Motivations

The main motivations for this paradigm are the possibility to benefit from the large literature in neural network computation, along with the inherent complexity and nonlinear mappings computed in an AE, and the flexibility allowed by the framework (large panel of activation functions, of architectural choices, etc).

In that sense, we hope, with Single-Song AutoEncoding, to improve the previously presented structural segmentation results, and to be able to extend the unsupervised compression paradigm with constraints and, in potential future work, supervision.

5.4.2 Paradigm Description

Practically, given a generic entry $x \in \mathbb{R}^n$, an AE learns a nonlinear function f with parameters θ (weights and biases of the network) such that $\hat{x} = f(x, \theta) \in \mathbb{R}^n$ reconstructs x as faithfully as possible. This is achieved by minimizing a given loss function $d(x, \hat{x})$. As for NTD and NMF, AE presented here can minimize the Euclidean distance (generally called "mean-squared error" in neural network settings), and the Kullback-Leibler and Itakura-Saito divergences [3].

An autoencoder is divided into two parts: an encoder, which compresses the input $x \in \mathbb{R}^n$ into a latent representation $q = f^e(x, \theta^e) \in \mathbb{R}^{B'}$ of smaller dimension (generally,

3. They can actually minimize many other differentiable loss function, but we restrict the study to these three loss functions.

$B' \ll n$), and a decoder, which reconstructs $\hat{x} = f^d(q, \theta^d)$, from q. A shallow encoder is constructed with one layer of weights R^e, bias b^e, and a nonlinear activation function σ, such that $q = \sigma(R^e x + b^e)$. The decoder follows as $\hat{x} = \sigma(R^d q + b^d)$.

Deep AE use the same formalism, but by stacking several layers; that is, for an encoder with l layers, $q = \sigma_l(R^e_l\, \sigma_{l-1}(R^e_{l-1}\, \sigma_{l-2}(...(R^e_1 x + b^e_1)) + b^e_{l-1}) + b^e_l)$. In general, the decoder mimics the architecture of the encoder, that is, if the i^{th} layer of the encoder R^e_i has sizes $k \times k'$, then the i^{th} layer of decoder R^d_i has sizes $k' \times k$. We apply this strategy in what follows.

As presented earlier, while AEs generally learn a common latent representation for an entire dataset, our technique consist of optimizing a network for each song. This framework is called Single-Song AutoEncoding, hence resulting in Single-Song AutoEncoders (SSAE).

> Single-Song AutoEncoding consists of optimizing an SSAE on every barwise spectrogram X by minimizing $d(X, \hat{X})$. The rationale is to study the different barwise latent representation $q \in \mathbb{R}^{B'}$, which are barwise compressed representation. Hence, an SSAE is optimized on a collection of barwise spectrograms $\{X_b, 1 \leq b \leq B\}$ for a song, resulting in a collection of barwise latent representation $\{q_b\}$, which forms a matrix $Q \in \mathbb{R}^{B \times B'}$, finally used for structural segmentation via an autosimilarity matrix and the CBM algorithm.

The reconstruction $\hat{X}$ is used for optimization purpose, and is of no use in the structural segmentation context. As for PCA, this Q matrix being potentially negative, we do not expect interpretable pattern to be obtained from the compression (here, patterns could be obtained from the decoder).

Depending on the architecture of the network, X can be a matrix in $\mathbb{R}^{F \times S}$ or a vector in $\mathbb{R}^{FS}$. The process is presented in Figure 5.8 with vectors as inputs and outputs of the network.

5.4.3 Network Architecture

The architecture of the network (*i.e.* the design of both encoder and decoder) has a major impact on the latent representations. In this thesis, layers are of two types: fully-connected and convolutional.

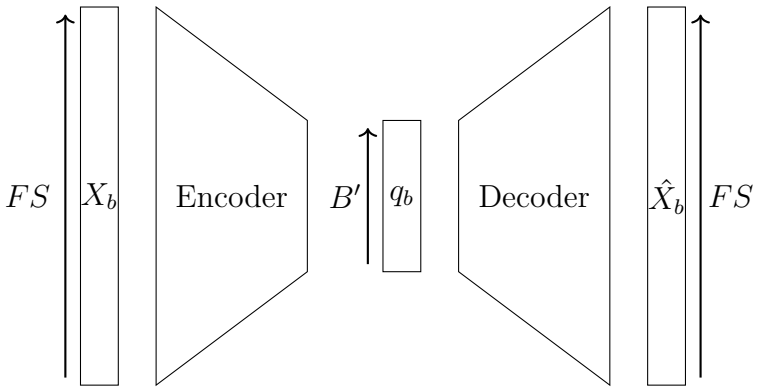


Figure 5.8 – Schematic representation of the general SSAE.

Convolutional layers lead to impressive results in image processing due to their ability to discover local correlations (such as lines or edges), which turn to higher-order features with the depth of the network [LeC+98] [GBC16, Ch. 9]. While local correlations are less obvious in spectrogram processing [PR21], Convolutional Neural Networks (CNN) still perform well in MIR tasks, such as MSA [GS15b].

In that context, we studied two neural networks, namely "FC" and "Conv" AutoEncoders. Though, this section only presents the Conv AutoEncoder, and abusively use the term of SSAE to denote the Conv AutoEncoder. This Conv SSAE is represented in Figure 5.9. The "FC" AutoEncoder is presented and studied in Appendix B only, mostly because of its poor performance.

The Conv SSAE architecture is designed as a trade-off between "shallow" and "deep" networks. In our paradigm, the number of samples on which the network is optimized is equal to the number of bars B, which, in RWC Pop, is on average 115 bars per song.

Hence, while the current trend in neural network computation leans towards deeper networks [PR21] (*i.e.* networks with large numbers of layers, such as VGGish [Her+17]), which are able to learn complex nonlinear functions, their achievements are also due to the processing of large datasets (for instance, the Million song dataset [Ber+11], AudioSet [Gem+17], or SALAMI used for training in [GS15b]), allowing to generalize across the dataset and adequately fit the large number of parameters induced by the large number of layers.

In our context, the number of samples is rather small, and largely smaller than the number of parameters of any deep network. On the other hand, PCA and AE are competitive when the AE is too shallow [Roc+18]. Thus, we tried to design networks large enough to compete with PCA, but small enough for the optimization paradigm to make

sense. The single-song optimization can still be seen as "overfitting" this particular song.

Overall, the details of the architecture are set quite empirically, following existing architectures. We use the nowadays quite standard Rectified Linear Unit (ReLU) function as the activation function, $\mathrm{ReLU}(x) = \max(0, x)$. This activation function is implemented in the output of each hidden layer, except in the last layer of the encoder, because it can result in null latent representations.

Layer Specifications

Encoder The encoder is composed of 5 hidden layers: 2 convolutional/max-pooling blocks, followed by a fully-connected layer, controlling the size B' of the latent space. Convolutional kernels are of size 3x3, and the pooling is of size 2x2, as in VGGish network [Her+17]. Convolutional layers define respectively 4 and 16 feature maps.

Decoder The decoder is composed of 3 hidden layers: a fully-connected layer (transpose of the previous one) and 2 "transposed convolutional" layers of size 3x3 and stride 2x2. A transposed convolution is similar to the convolution operation taken in the backward pass: an operation which takes one scalar as input and returns several scalars as output [DV16, Ch. 4]. Hence, it is well suited to reverse the convolution process.

Batch Normalization Layers Recent neural networks architectures generally contain batch normalization layers [IS15]. A batch normalization layer consists of two steps: firstly, normalizing the batch values for them to be of unit variance and zero mean, and, secondly, rescaling these values with an affine transformation of learnable parameters.

Denoting as x a sample in the batch, μ and σ respectively the average and standard deviation of the values in this batch, the batch normalization layer applies the transformation:

$$\mathrm{bn}(x) = \left(\frac{x - \mu}{\sigma}\right) s + m \tag{5.2}$$

where parameters s and m are trained in the optimization process, and can be seen as new standard deviation and average for the outputs of the batch normalization layer.

Batch normalization layers often lead to a gain in performance, while speeding up the training process [IS15]. Indeed, by normalizing the input of each layer, the optimization process is less dependent on the distribution of layer's weights, which may vary with the different epochs and optimization steps, especially in the first epochs due to the random initialization. Hence, batch normalization layers can stabilize the optimization process

with respect to the scaling of weights. In addition, the use of batch normalization layers is motivated in our context as reducing the parameter space.

Practically, batch normalization layers are implemented for each hidden layer, before the input and after the pooling layer. The impact of batch normalization layers on structural segmentation results is presented in Appendix B.3, which presents improved segmentation results when using such layers.

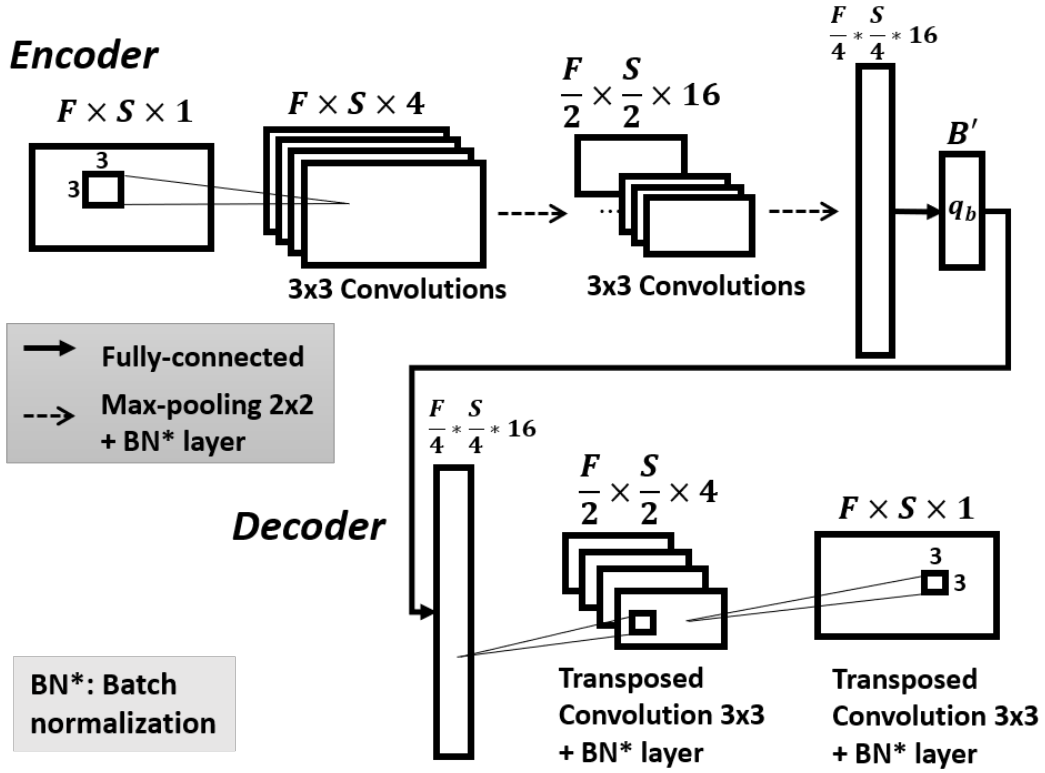


Figure 5.9 – Architecture of the Conv SSAE.

Initialization

The initialization settings are of primordial importance, and largely influence the optimization process and output of the network. Note though that this is also true for NMF [BG08; Gil20] and, subsequently, NTD, but, for both methods, we found consistent and deterministic initialization strategies (NNDSVD [BG08] for NMF and High Order SVD [DDV00] for NTD).

SSAEs are initialized following the uniform distribution defined in [He+15], also known as "kaiming" initialization. SSAEs were introduced in [MCB22c], and the associated ex-

perimental results were obtained by fixing the pseudo-randomness of the distribution with a seed.

> In this study, we compare experimental results obtained from five different pseudo-random initializations (all following the "kaiming" distribution), fixed with five different seeds.

We believe that this strategy better reports the real potential of SSAEs, by aggregating the results on several computations. In particular, for each initialization, the average over the whole dataset is computed for each metric (as for the other methods). The results are then presented in boxplots as the median, Q1 and Q3 quartiles, and extremums of these five averages.

When a unique percentage is presented for an SSAE (for instance in figures), this percentage is the median of the five runs. When the median is presented with a margin (*e.g.* $10\% \pm 1\%$), this margin represents the Median Absolute Deviation (MAD). The MAD is the equivalent of the standard deviation for the median: denoting as $\text{med}(x)$ the median of a set of values x_i, $\text{MAD}(x) = \text{med}(|x_i - \text{med}(x)|)$.

Implementation Details

SSAEs are developed using *Pytorch* 1.8.0 [Pas+19], trained with the Adam optimizer [KB14], with a learning rate of 0.001, divided by 10 when the loss function reaches a plateau (20 iterations without improvement) until 1e-5. The optimization stops if no progress is made during 100 consecutive epochs, or after a total of 1000 epochs.

An important parameter in the optimization process is the size of the batch when processing each song, *i.e.* the number of samples to be presented to the network before backpropagating the error and updating the network' parameters. It is consensual in neural network computation to split the dataset in several batches, one of the main reasons being memory issues, as a large dataset cannot fit at-once. Memory issues are not a concern here, as we deal with each song separately.

Still, it is considered beneficial optimization-wise (for generalization purposes) to update the neural network' parameters in different batches [WM03], resulting in several updates of the network per epoch. In our context, generalization is not an objective, and processing each song in several batches leads to longer optimization schemes (in terms of computation time).

Results presented in [MCB22c] were reported with a batch size of 8 on the RWC Pop dataset, which was chosen empirically. In this thesis, we studied the impact of different batch sizes in Appendix B.2, and concludes towards the processing of each song in unique batches, *i.e.* backpropagation performed on the entire song at-once.

Hence, the song is processed in a unique batch. Therefore, the aforementioned "batch normalization" layers actually refers to the normalization of the entire input (*i.e.* the song). For the simplicity of notation and convention, we keep the name "batch normalization", even if not accurate for our context.

Optimizing an SSAE is quite heavy computationally, in particular compared to NMF or PCA. As an example, on an Intel® Core(TM) i7 CPU, decomposing the song *POP01* with $B' = 16$ for the Nonnegative Log Mel (NNLM) spectrogram takes approximately 6 and a half minutes for the Conv SSAE [4].

As for previous experiments with NMF and PCA, we focus on the RWC Pop dataset for hyperparameter tuning, which contains fewer songs. In addition, we decided to fix the size of the latent space. It is expected that the size of the latent space influences segmentation results, as for the other techniques. Still, we believe that the impact of the different sizes of the latent space would collide with the impact of the different initializations. AE are frequently compared with PCA due to their proximity [Roc+18], and can even be designed so as to reproduce the PCA [OK85; BH89]. In that sense, we choose the dimension which was most frequently picked by cross-validation in the previous experiments (Section 5.3.3), which is 24 [5].

5.4.4 Structural Segmentation Experiments

Similarly to previous paradigms (NTD, NMF and PCA), barwise compressed representation are used for structural segmentation, via the Q matrix and the CBM algorithm. An example of Cosine autosimilarity is presented on Figure 5.10.

Experiments are designed to answer the three following questions:

Question 10 *How are segmentation performance impacted by the similarity function (Cosine, Covariance and RBF) with the CBM algorithm applied to SSAE-based autosimilarities?*

4. In general, neural networks are faster when run on GPU instead of CPU, and SSAEs are indeed faster when run on GPU (in the order of magnitude of seconds), but we present results on CPU for fair comparison with the other techniques.

5. Two-thirds of the ranks selected are either 24 or 40, for all similarities, with an advantage for 24.

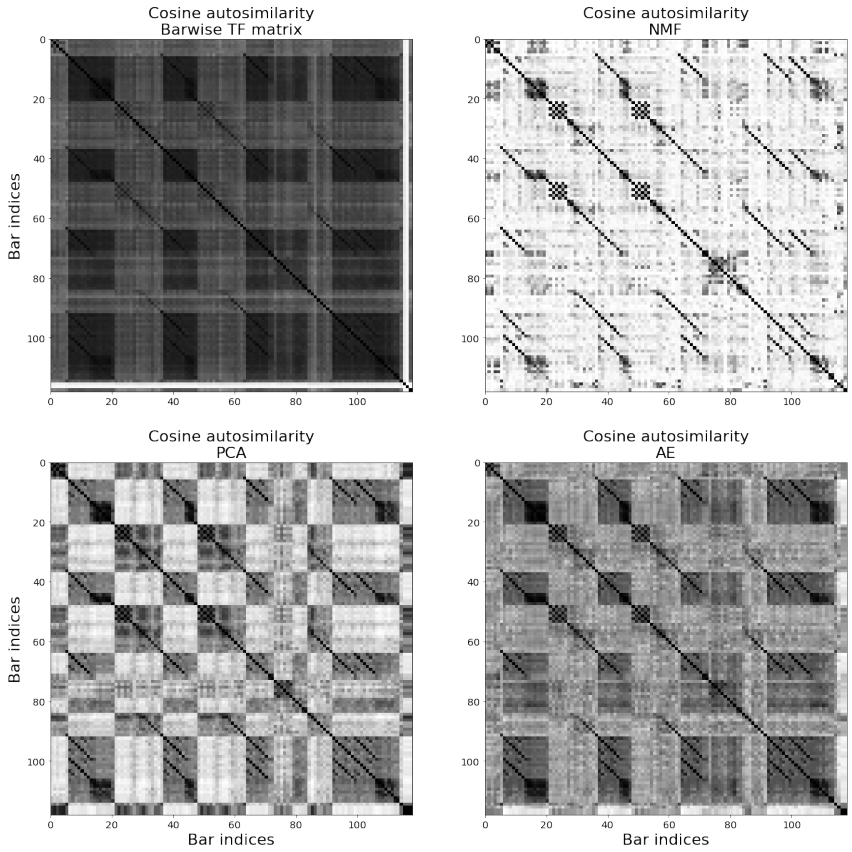


Figure 5.10 – Example of Cosine autosimilarity of the SSAE, compared with the Barwise TF-, NMF- and PCA-based Cosine autosimilarities. These autosimilarities are computed on the NNLMS of the song *Pop01* of RWC Pop, with $B' = 24$.

Question 11 *How is the SSAE-based Cosine similarity performing compared to all previously presented Cosine autosimilarities (i.e. Barwise TF-, NTD-, NMF- and PCA-based autosimilarities) in the structural segmentation task, for all features?*

Question 12 *What is the impact of the loss function (i.e. Euclidean loss, KL- and IS-divergences) on segmentation results when the feature exhibits energy discrepancies between frequencies (such as for STFT and Mel spectrograms) and when the energy discrepancies between frequencies are mitigated in the feature (such as energy-normalized chromagrams)?*

The first experiments focus on the RWC Pop dataset, leading to initial experimental conclusions (notably towards the best-performing features), later and further studied on the SALAMI dataset for generalization on a larger and more diverse dataset.

The SSAEs are subject to different loss functions, namely the Euclidean distance, and the KL- and IS-divergences, respectively defining the "Euclidean-SSAE", the "KL-SSAE" and the "IS-SSAE".

RWC Pop Dataset

Figure 5.11 presents segmentation results computed with the Euclidean-SSAE, with the three similarity functions and the different features. Cosine and Covariance autosimilarities obtain very similar results for both $F_{0.5}$ and F_3.

Except for the Chromagram, the RBF autosimilarity obtain lower results at $F_{0.5}$ and similar results at F_3 than the ones obtained with Cosine and Covariance autosimilarities. On the chromagram, the best result is obtained for the RBF autosimilarity. Still, as the best result with the chromagram is lower than results obtained with the Log Mel, NNLM and MFCC spectrograms, we focus future experimentation on the Cosine autosimilarity only. Note that a similar trend was observed for PCA.

Figure 5.12 presents segmentation results obtained with all barwise compression techniques, when optimized subject to the Euclidean distance (*i.e.* the Barwise TF, result of the CBM algorithm with no barwise compression, and the previous barwise compression techniques: NTD, NMF and PCA).

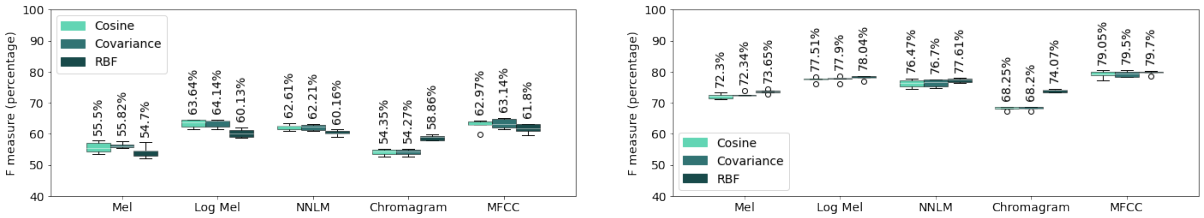
In these results, the Euclidean-SSAE is performing similarly to PCA, in particular for the $F_{0.5}$ metric, which is not an obvious conclusion; but, conversely to results presented in [MCB22c], the best median score for the SSAE does not outperform the best score of PCA, for both $F_{0.5}$ and F_3. Results presented in [MCB22c] were obtained using one particular initialization and keeping the best of five latent spaces dimensions, while the current results aggregate five different initializations and use a unique latent space dimension. We assume that this condition leads to a loss in performance, but better represents the potential of the SSAE.

Figure 5.13 compares segmentation results obtained with the Euclidean-, KL- and IS-SSAEs. As a first conclusion, the IS-SSAE consistently obtain worse results than both other SSAEs, which is different than previous compression methods. We do not explain this result, but we notice that this is consistent with findings in [NLV16], where the IS-divergence is the worst performing loss function, notably compared to both Euclidean distance and KL-divergence.

In addition, in contrast with the other barwise compression techniques, the Euclidean- and KL-SSAEs obtain similar results with both NNLMS and chromagrams, while we

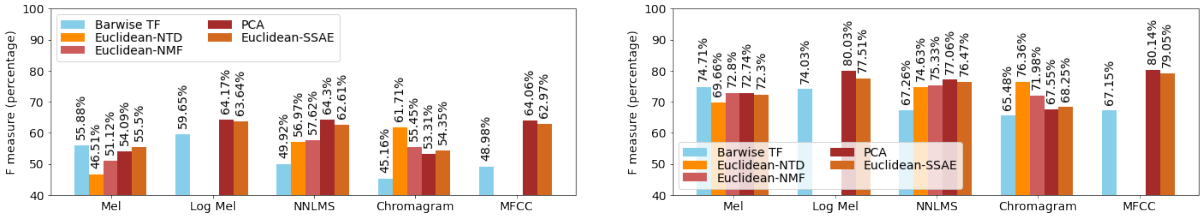
expected significant differences in segmentation results. Differences only appear for the Mel spectrogram, where the KL-SSAE outperforms the Euclidean-SSAE, as expected with results obtained with the other barwise compression techniques. We do not explain why the KL-divergence is less advantageous for the SSAEs, compared to NTD and NMF.

Finally, Table 5.3 presents the best segmentation results for the different techniques. In the best of conditions, the SSAE outperforms both NMF and NTD at $F_{0.5}$, but obtains similar performance at F_3. In addition, the best SSAE do not achieve the best performance obtain with PCA and with the RBF autosimilarity of the Barwise TF.



(a) $F_{0.5}$. (b) F_3.

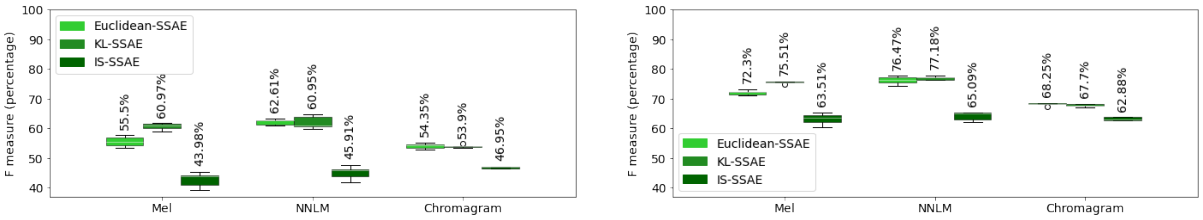
Figure 5.11 – Segmentation results according to the feature and the similarity function, for the Euclidean-SSAE. The scores above the boxplots represent the median of averaged F measures according to the five different runs.



(a) $F_{0.5}$. (b) F_3.

Figure 5.12 – Median segmentation results according to the feature, for the Cosine autosimilarity of the Euclidean-SSAE and the previous barwise compression techniques.

In light of these results, it seems that the Cosine autosimilarity is a good choice for the similarity computation of SSAE-compressed representations, and that Euclidean- and KL-SSAEs are relatively similar in term of segmentation performance. In addition, the SSAE obtain results close to those of PCA, which is the best compression method segmentation-wise until now.



(a) $F_{0.5}$. (b) F_3.

Figure 5.13 – Segmentation results according to the nonnegative features, for the Cosine autosimilarities of the Euclidean-, KL- and IS-SSAEs.

Method	Best conditions	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Barwise TF	Log Mel, RBF	**64.32%**	**70.00%**	**66.52%**	78.31%	**85.64%**	**81.16%**
NTD	NNLM, KL, Cosine	60.34%	63.84%	61.79%	78.25%	83.32%	80.37%
NMF	NNLM, KL, Cosine	61.21%	62.99%	61.64%	**79.02%**	81.61%	79.75%
PCA	Log Mel, Cosine	61.86%	67.78%	64.17%	76.96%	84.73%	80.03%
Conv SSAE	MFCC, Euclidean, Covariance	60.58% ±0.84%	67.15% ±2.28%	63.14% ±2.25%	76.66% ±0.26%	84.09% ±1.18%	79.50% ±1.64%

Table 5.3 – Comparison of the SSAEs with the best-performing methods, on RWC Pop.

SALAMI Dataset

Starting from the previous conclusions, obtained on the RWC Pop dataset, SSAEs are studied on MFCC and NNLM spectrograms. Segmentation results, computed with the different similarity functions, are presented on Figure 5.14.

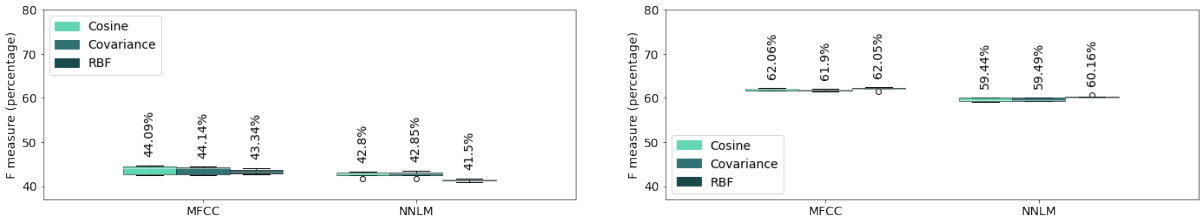
As for the RWC Pop dataset, results are similar across the different autosimilarities. Table 5.4 presents results for the Cosine autosimilarity for the Euclidean-SSAE on the MFCC, and for the Euclidean-, KL- and IS-SSAEs for the NNLMS.

Results on the SALAMI dataset lead to similar conclusions than for the RWC Pop dataset: the IS-SSAE is performing worse than the other networks, and the Euclidean- and KL-SSAE obtain similar results. The best results are obtained on the MFCC feature.

Conclusions for the SSAE

Finally, we conclude towards the use of SSAE for structural segmentation by answering to the aforementioned questions.

Experimental conclusion 10 *With the current version of the CBM algorithm, the choice of the similarity function does not significantly impact the segmentation performance of*



(a) $F_{0.5}$. (b) F_3.

Figure 5.14 – Segmentation results according to the different similarity functions features, for the Euclidean-SSAE, on the NNLM and MFCC spectrograms.

Feature	Method		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
MFCC	Barwise TF*		35.97%	34.47%	34.13%	53.48%	51.26%	50.79%
	SSAE		**38.49%** ±1.65%	**54.20%** ±0.71%	**44.09%** ±0.84%	**53.67%** ±0.91%	**78.09%** ±1.57%	**62.06%** ±0.26%
NNLM	Barwise TF*		38.86%	41.63%	39.93%	**58.45%**	62.68%	58.63%
	PCA		36.12%	57.75%	43.06%	50.54%	81.57%	60.45%
	NMF (KL)		**39.01%**	52.11%	**43.42%**	55.41%	74.53%	**61.84%**
	SSAE	Euclidean	35.53% ±0.40%	**58.23%** ±0.58%	42.80% ±0.36%	49.25% ±0.40%	81.82% ±1.40%	59.44% ±0.45%
		KL	35.50% ±0.36%	58.21% ±0.77%	42.74% ±0.39%	49.44% ±0.28%	**81.97%** ±0.90%	59.77% ±0.45%
		IS	27.75% ±0.98%	49.13% ±1.42%	34.46% ±0.11%	42.16% ±1.35%	77.43% ±0.29%	52.88% ±1.21%

Table 5.4 – Segmentation results of SSAEs on SALAMI with the Cosine autosimilarity, compared to the best other segmentation methods. *The Barwise TF condition means that there is no compression.

the SSAE.

Experimental conclusion 11 *Reported segmentation results show, in every condition, an advantage of the SSAE-based Cosine autosimilarity over the Barwise TF-based Cosine autosimilarity.*

In most of conditions, SSAEs obtain similar results than PCA, and better or similar results than both NTD and NMF.

A limit to the comparison with the other compression methods in **Experimental conclusion 11** is that B' is fixed to 24 for the SSAE, when it was fitted by cross-validation for the other techniques. Hence, we can assume that the SSAE is penalized by this restriction.

Experimental conclusion 12 *Both Euclidean distance and KL-divergence, when used as loss function for the SSAE, perform similarly on both datasets for NNLMS and chromagrams. A slight advantage appears for the KL-SSAE over the Euclidean-SSAE on Mel spectrograms, but this score remains lower than scores obtained on NNLMS. The IS-SSAE is significantly performing worse than both other SSAEs.*

Experimental conclusions 10 and **11** are in line with the conclusions obtained with NTD, NMF and PCA, *i.e.* that compression-based autosimilarities vary less according to the choice of the similarity function, and that the Cosine autosimilarity of compressed-based representations are better performing than the raw Cosine autosimilarity of the Barwise TF.

Still, conversely than with the other techniques (NTD and NMF), **experimental conclusion 12** states that using either the Euclidean distance or the KL-divergence as loss function does not influence segmentation performance with the NNLMS and chromagrams, while the IS-SSAE obtains worse performance. Hence, compared to NTD and NMF, it is less important for the SSAE to adapt the loss function to the characteristics of the feature.

5.4.5 Additional Experiments: Extending the SSAEs

In addition to the structural segmentation experiments of Section 5.4.4, evaluated on the whole RWC Pop and SALAMI datasets, this Section presents one-song experiments, showcasing the potential of SSAE with some architectures modifications.

These experiments are computed on the Log Mel spectrogram of the song *POP01* of RWC Pop. The rationale of these experiments is to present two additional paradigms, that are perspectives we have not had the time to fully explore.

Activation Function on the Latent Space

The first idea consists of adding an activation function on the latent space.

> Adding an activation function on the latent space constrains the values of the Q matrix.

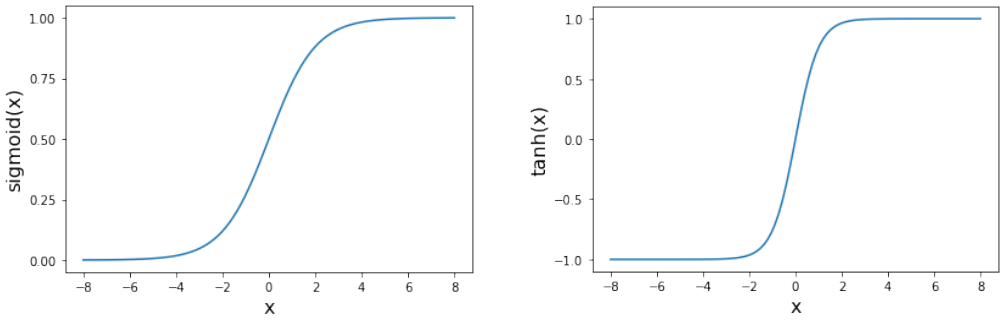
The different activation functions are motivated as specific constraints on the entries of matrix Q. In this paradigm, as the batch normalization layer prior to the latent space

already constrains values towards a certain distribution (which is learned in the optimization process), the activation function is used in replacement of the batch normalization layer on the latent space, while the other batch normalization layers remain unchanged.

— Softmax: the Softmax function, for a vector x, is applied to all of its values x_i as $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$. This function rescales the values x_i such that $x_i \in [0, 1]$ and $\sum_i x_i = 1$, while the exponential function favors the highest values in x. This is the sense of the name "softmax", as the highest value is exponentially increased, squashing the other values.

— Sigmoid: $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. The Sigmoid function constrains values to be taken in the range $[0, 1]$, and particularly to smoothly quantize values to 0 and 1: with the exponential function, negative values for x quickly fade to values close to 0 in the sigmoid, while positive values for x quickly result in values close to 1 in the sigmoid. The Sigmoid function is presented in Figure 5.15a.

— Hyperbolic Tangent (Tanh): $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x}-1}{e^{2x}+1}$. The hyperbolic tangent takes values in $[-1, 1]$, and $\tanh(0) = 0$. As for the previous functions, the exponential aims at quickly pushing values towards the extremums, *i.e.* smoothly quantizing values to 1 and -1. The hyperbolic tangent function is presented in Figure 5.15b.



(a) Sigmoid function.

(b) Hyperbolic tangent function.

Figure 5.15 – Plots of both Sigmoid and hyperbolic tangent functions.

Figures 5.16, 5.17, 5.18 and 5.19 present the Q matrices obtained with the different latent activation functions (respectively no activation function, the Softmax, the Sigmoid and the hyperbolic tangent), and their respective Cosine autosimilarities. These matrices are computed from the Euclidean-SSAE with a latent space of dimension 16 (for better visualizations, leading to smaller Q matrices than the dimension of 24 chosen in Section 5.4.4).

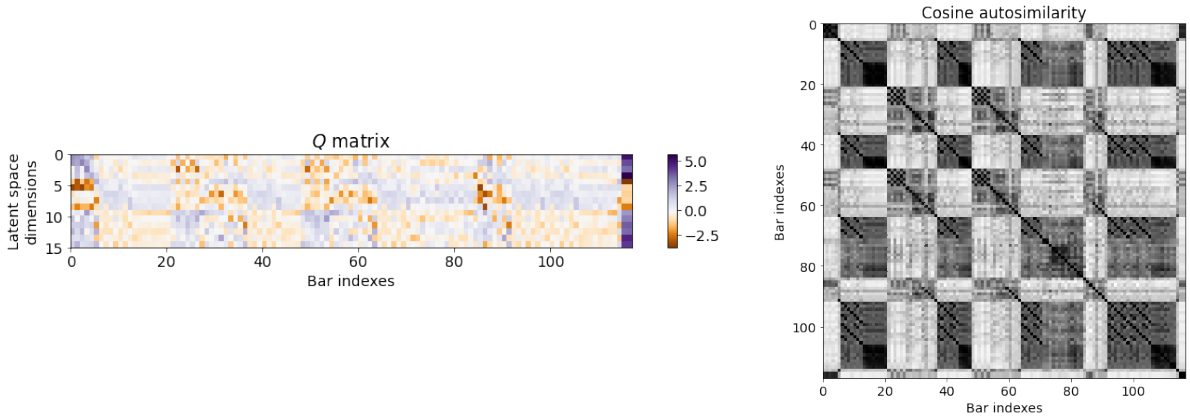


Figure 5.16 – Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, without activation function on the latent space, for the Log Mel spectrogram of the song *POP01*.

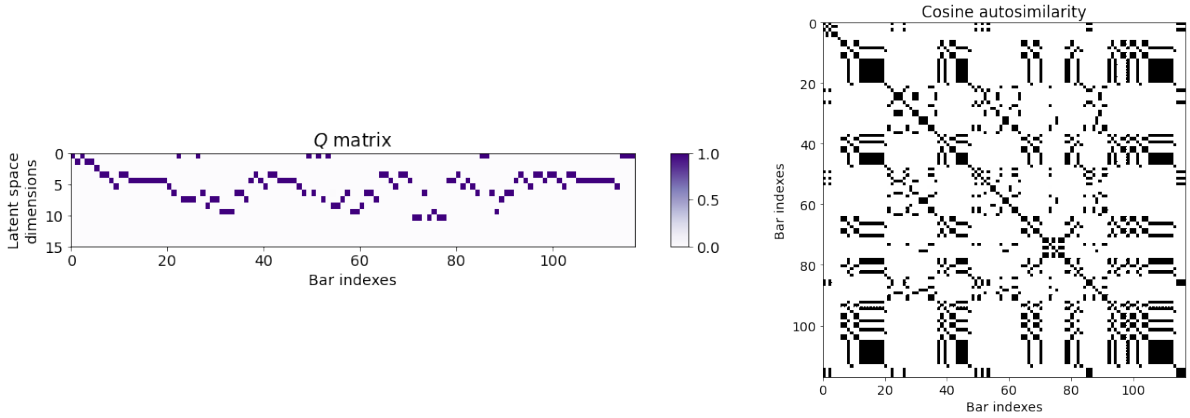


Figure 5.17 – Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, with the Softmax activation function on the latent space, for the Log Mel spectrogram of the song *POP01*.

Visually, it seems that the activation functions are fulfilling their role by pushing values towards extremums. In particular, using the hyperbolic tangent as activation function could particularly be useful for structural segmentation, providing a high contrast between similar and dissimilar bars.

With the Softmax activation function, bars are represented with only one coefficient, which weakens the homogeneity in the autosimilarity matrix, hence deteriorating the potential of the CBM algorithm in retrieving the structure of the song. Still, the Softmax could be interesting for pattern uncovering and musical analysis, provided an interpretable decoder, as it extracts the most important pattern in this bar.

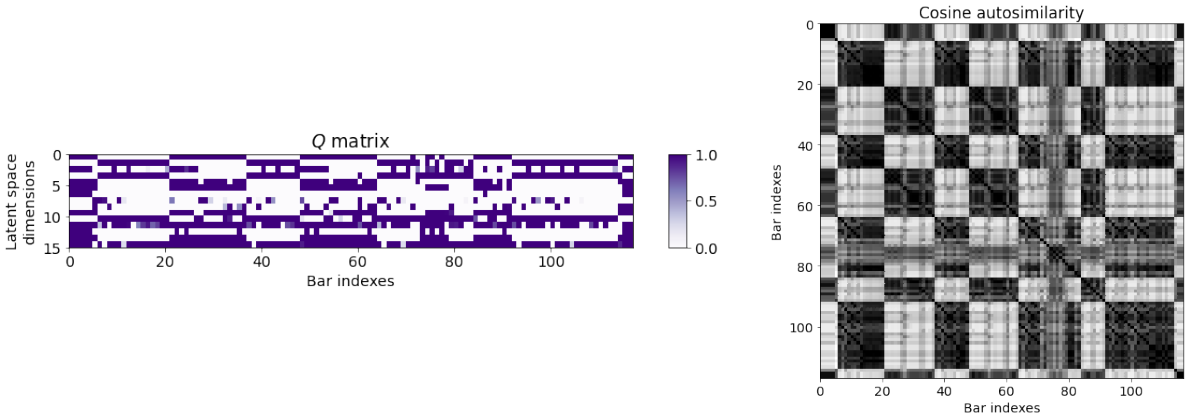


Figure 5.18 – Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, with the Sigmoid activation function on the latent space, for the Log Mel spectrogram of the song *POP01*.

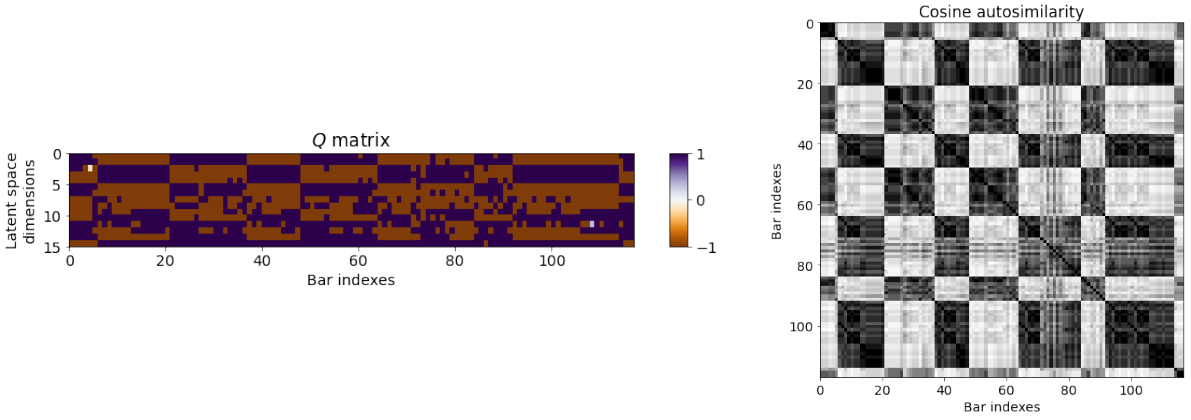


Figure 5.19 – Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, with the hyperbolic tangent activation function on the latent space, for the Log Mel spectrogram of the song *POP01*.

The Sigmoid activation function could be useful in both these regards, *i.e.* for enhancing the homogeneity in the autosimilarity and the interpretability of patterns.

Future work could further investigate this paradigm, and study other activation functions or constraints.

Triplet Loss

The previously presented SSAE is totally blind, *i.e.* not informed with notion of structure or similarity in the song prior to the optimization. Informing the network about a notion of similarity between segments could be a great asset. In particular, this is the

heart of the method developed by McCallum [McC19], using the idea that beats which are close probably belong to a same segment.

In our case, the rationale is that the Cosine autosimilarity of the Barwise TF representation (*i.e.* uncompressed) already catches the main similarities in the song, but fails at finding relevant dissimilarity. Hence, we focus on increasing the contrast between similar and dissimilar segments, by informing the network with prior information of similarity and dissimilarity.

Our strategy, as in [McC19], is based on the Triplet Loss [SKP15]. The Triplet Loss is an architecture paradigm used in metric learning where, instead of optimizing the network directly on the task with supervision (in the original article, person identification), the network is optimized on data triplets, with the goal to learn an embedding where the distance between data points is used for the task in a later stage.

Hence, supposing that the data triplets are provided (which can be based on labels, as in [SKP15], or on data information/priors, as in [McC19]), the process is fully unsupervised. In details, the Triplet Loss $L_T(a, p, n)$ is based on three examples:

— The anchor a, which is the current data point considered,
— The positive example p, which should be close to the anchor in the embedding space,
— The negative example n, which should be far from the anchor in the embedding space.

Given these examples, and denoting as $f(x)$ the projection of x in the embedding space, L_T is defined in Equation 5.3:

$$L_T(a, p, n) = \left[\|f(a) - f(p)\|_2^2 - \|f(a) - f(n)\|_2^2 + \alpha\right]_+ , \tag{5.3}$$

where α is an hyperparameter, called "margin".

Practically, the Triplet Loss is 0 if the distance between the anchor and the positive example is smaller than the distance between the anchor and the negative example plus a margin (*i.e.* the positive example is closer with at least a distance α to the anchor than the negative example). Otherwise, the loss function increases, penalizing the optimization paradigm, which should bring the positive example closer to the anchor and/or push the negative example further away in the subsequent iterations. Parameter α is generally set to 1.

In our context, the idea is to compute relevant similarity and dissimilarity notions between latent representations. The Triplet Loss may be useful in that regard, constraining the distance between the different latent representations, hence applied on a triplet of latent representations.

This defines a "Triplet SSAE", mixing both the SSAE and the Triplet Loss paradigms, resulting in a network optimized on both reconstruction and Triplet Loss, as in [Oku+17].

Practically, given a triplet of barwise spectrograms (X_b^a, X_b^p, X_b^n), and denoting as (Q_b^a, Q_b^p, Q_b^n) and $(\hat{X}_b^a, \hat{X}_b^p, \hat{X}_b^n)$ respectively their latent representation and their reconstruction, the Triplet SSAE is optimized via Equation 5.4. A schematic Triplet SSAE is presented in Figure 5.20.

Reconstruction errors — Triplet Loss

$$\arg\min \; d(X_b^a, \hat{X}_b^a) + d(X_b^p, \hat{X}_b^p) + d(X_b^n, \hat{X}_b^n) + L_T(Q_b^a, Q_b^p, Q_b^n) \tag{5.4}$$

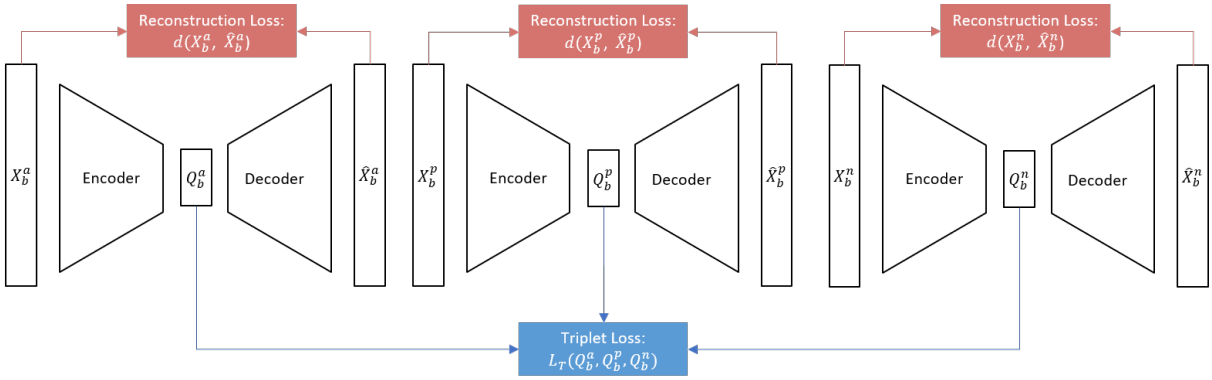


Figure 5.20 – Schematic representation of the Triplet SSAE.

The paradigm is highly sensitive to the definition of the triplets. In our context, positive and negative samples should be selected based on structural information. In our paradigm, though, based on single-song analysis, informing about the structure of the song (*i.e.* supervising the learning scheme) would be a huge bias, not counteracted by generalization ability.

In that sense, we rather opt for an unsupervised technique for the generation of triplets, based on the feature-wise similarity of the song. Indeed, our main idea with the triplet loss being the increase of contrast between similar and dissimilar bars compared to the feature-wise similarity, we compute positive and negative examples by thresholding the initial similarity distribution in the Barwise TF matrix. For each bar, seen as the anchor:

— A positive example is randomly sampled from the 10% most similar bars,

— A negative example is randomly sampled from the 50% least similar bars.

The similarity in "most/least similar bars" is the Cosine similarity between bars of the Barwise TF, and these thresholds are chosen empirically. In addition, the number of samples (bars per song) being relatively small, triplets are regenerated at each epoch.

With this strategy, Figure 5.22 presents the optimization result for the song *POP01*. It seems, empirically, that the network indeed results in more contrasted outputs, compared to Figure 5.21, which presents the Q matrix computed from the Euclidean-SSAE.

In particular, the values of the Q matrix obtained with the triplet loss are more concentrated on the extremums than those of the Q matrix of the original SSAE, favoring the homogeneity.

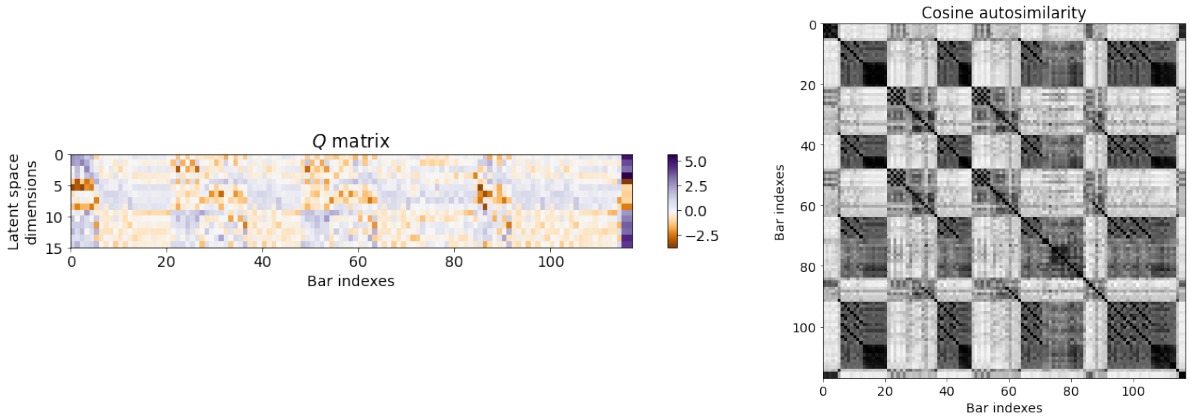


Figure 5.21 – Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, without activation function on the latent space, for the Log Mel spectrogram of the song *POP01* (shown again for comparison with the Triplet Loss model).

Hence, assuming that the similarity of the Barwise TF holds the adequate structural information in its self-similarity, only not contrasted enough, the triplet loss may be an interesting paradigm to enhance the contrast.

Conversely, this method introduces additional hyperparameters (the margin α, the thresholds for the generation of triplets, and, potentially, a ponderation parameter for

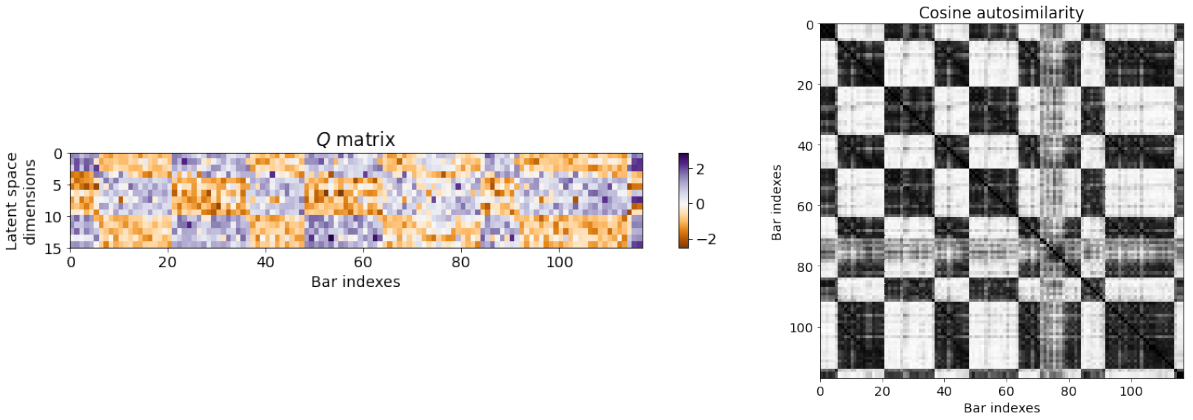


Figure 5.22 – Q matrix and Cosine autosimilarity of the latent representation of the Triplet Conv SSAE, optimized subject to the Euclidean distance, for the Log Mel spectrogram of the song *POP01*.

mitigating between both losses, not introduced in Equation 5.4) which must be tested and set.

In addition, it increases the computation time, in this test by a factor three: each example being a triplet, each sample is composed of three spectrograms to optimize instead of one.

Overall, we believe that this strategy is a promising paradigm.

5.5 Conclusions

This section was dedicated to the presentation and the study of three barwise compression schemes: Nonnegative Matrix Factorization (NMF), Principal Component Analysis (PCA) and Single-Song AutoEncoders (SSAE). These compression schemes were then used in the context of structural segmentation, by studying the autosimilarities of the respective barwise compressed representations.

In general, PCA and SSAE obtain similar or better results than NTD and NMF, especially when optimized subject to the Euclidean-distance, and we conclude on experimental results that there exists a trade-off between the interpretability of solutions (NTD/NMF) and better segmentation results (PCA/SSAE), which we will study in more details in Section 6. In addition, it seems that using the KL-divergence is more an advantage for the NTD than for the SSAE, and that the IS-divergence is a disadvantage for the SSAE, but we are not able to explain this result.

Table 5.5 summarizes the best experimental results of these techniques, and compares the compression-based results with the State-of-the-Art techniques. In short, all barwise compression techniques obtain similar results in the best of their conditions, except for $F_{0.5}$ on the RWC Pop dataset. In this latter condition, both PCA and SSAE are better performing than NTD and NMF. All compression techniques obtain lower results than those of the State-of-the-Art CNN [GS15b] for $F_{0.5}$, but are competitive for F_3, and they all outperform the blind State-of-the-Art for both metrics.

In addition, the overall best results with the CBM algorithm is still obtained without compression, with the RBF autosimilarity. Overall, compressed representations obtain more robust results with respect to the similarity function than uncompressed representations with the CBM algorithm, indicating a better contrast between similar and dissimilar bars. In addition, all experiments were conducted using the CBM algorithm whose parameters (kernels and penalty function) were optimized according to the RBF autosimilarity of the Barwise TF matrix, and it could bias the results.

Method	Best conditions	RWC Pop		SALAMI	
		$F_{0.5}$	F_3	$F_{0.5}$	F_3
Barwise TF	Log Mel, RBF	66.52%	**81.16%**	44.94%	**63.18%**
NTD	NNLM, KL, Cosine	61.79%	80.37%	-	-
NMF	NNLM, KL, Cosine	61.64%	79.75%	43.42%	61.84%
PCA	Log Mel, Cosine	64.17%	80.03%	43.55%	61.79%
Conv SSAE	MFCC, Euclidean, Covariance	63.14% ±2.25%	79.50% ±1.64%	44.04% ±1.33%	61.91% ±0.55%
Foote [Foo00]		34.48%	55.01%	32.70%	54.23%
CNMF [NJ13]		28.81%	46.53%	27.29%	44.72%
Spectral Clustering [ME14a]		45.01%	60.30%	33.69%	48.44%
Structural Features [Ser+14]		42.96%	62.15%	33.84%	53.72%
CNN [GS15b]		**69.70%**	79.34%	**54.09%**	62.28%

Table 5.5 – Comparison of all best-performing methods.

In details, NMF is generally less performing than NTD, but is faster and requires to fix only dimension, compared to three for NMF. In addition, the NMF model is largely studied in literature (see for instance [Gil20]), and this literature could be exploited to further improve Barwise NMF (for instance by implementing constraints and/or developing different optimization paradigms).

Potential future improvements of the Barwise NMF model are those already outlined for NTD, both methods being conceptually similar. In addition, one could imagine a

mixed optimization paradigm, where three NMF performed on the different unfoldings of the TFB tensor could be used to initialize the three factors of NTD, which could reduce the overall number of iterations and hence the global computation time of NTD. We never tested this paradigm.

PCA is overall the best-performing model, and the fastest one. SSAEs obtain very similar results, still consistently lower when averaged over five runs, and requires much larger computation capacities. Nonetheless, results for the SSAEs are computed with a unique latent space dimension, when five were evaluated for the PCA (and similarly for NTD/NMF), which could favor the latter techniques.

In addition, the presented SSAE are easily improvable, which seems more complicated PCA. In particular, we presented two potential directions for improving SSAE, namely adding constraints on the latent space and adding prior knowledge with the Triplet Loss paradigm.

In addition, SSAE could benefit from the large literature in the domain, and in particular benefit from different constraints, prior knowledge and supervision, for instance via Transfer Learning [WKW16], *i.e.* using the first layers of a trained neural networks to benefit from the learned representations, and optimize the deepest layers for the current task.

In addition, some recent neural networks architectures were designed so as to disentangle several components of the inputs, for instance, in speech processing, disentangling language content, pitch and rhythm [Qia+20]. In our context, disentangling the rhythmic from the timbral and from harmonic/melodic information could be a real asset to better catch the changing parameters constituting the structure.

Chapter 6

AE-NTD

Synopsis

This chapter presents the AE-NTD, a paradigm mixing both NTD and AutoEncoders, studied on both structural segmentation and pattern uncovering tasks.

6.1 Introduction

This thesis has presented several barwise compression schemes, namely NTD, NMF, PCA and SSAE, and the associated barwise representations were used in the context of structural segmentation and, to some extent, pattern uncovering. Among the techniques, PCA and SSAE obtain the highest segmentation scores, while NTD and NMF are slightly less performing.

The main differences between these techniques are the nonnegativity constraints enforced in NTD/NMF and not present in PCA/SSAEs, which can provide interpretable outputs, as presented for NTD in Chapter 4. A trade-off between performance and interpretability seems to appear in light of these results, and this chapter aims at further investigating this trade-off.

In particular, this chapter focuses on both NTD and SSAE paradigms, and mixes them in a new paradigm called "AE-NTD". The rationale is that both NTD and AE are in some way based on matrix products, and that the structure of NTD can be implemented in the SSAE framework.

The objective with this new paradigm is to mix the high performance of SSAE with the interpretability of NTD, and, in addition, broaden the horizons of NTD with the neural network tools and literature.

The first part formally recasts the decoder of an AE in the NTD framework, and introduces the motivations behind this new paradigm. A second part is dedicated to the experimental study of the AE-NTD, notably regarding structural segmentation and pattern uncovering, and to compare the results of the previously introduced SSAEs with the AE-NTDs.

The contributions reported in this section are twofold:

— Formalism: while AEs have already been linked to matrix factorization models such as PCA [OK85; BH89], NMF [SV17], and even the CANDECOMP/PARAFAC tensor decomposition [CCS19], the current chapter is, to the best of our knowledge, the first attempt to recast the NTD in neural network formalism. Still, it is largely influenced by the aforementioned existing works.

— Experimental: mixing AE and NTD results in a new barwise compression scheme, taking advantage from both the interpretability of NTD and the high performance and flexibility of AE. In particular, the trade-off between interpretability and segmentation performance is observed in the experimental results.

Both the code and the conducted experiments relative to this section are included in the open-source *BarMusComp* toolbox [MCB22b].

6.2 Mathematical Formalism

The AE-NTD paradigm is a new barwise compression paradigm, imposing the structure of NTD in an AutoEncoder. In that sense, fundamentals of both the NTD and the SSAEs are presented hereafter, as reminders of previous sections. Then, by first introducing the AE-NMF model [SV17], we formally introduce AE-NTDs.

6.2.1 Reminder of NTD at the Barscale

NTD was defined in Chapter 4 as the multiway product $\mathcal{X} \approx \mathcal{G} \times_1 W \times_2 H \times_3 Q$, which can be rewritten $\mathcal{X}_{(3)} \approx Q\mathcal{G}_{(3)}(W \otimes H)^\intercal$ when unfolding on the third mode (*i.e.* focusing on the Q matrix). In this setting, matrix $\mathcal{X}_{(3)}$ is approximated as the product of three nonnegative matrices, Q, $\mathcal{G}_{(3)}$ and $(W \otimes H)^\intercal$.

It should be noted that the matrix $\mathcal{X}_{(3)}$ is exactly the Barwise TF matrix, where each row is a barwise spectrogram (*i.e.* a vector where both time and frequency dimensions are fused). For a particular bar b, the product becomes $\mathcal{X}_{(3)_{b:}} \approx Q_{b:}\mathcal{G}_{(3)}(W \otimes H)^\intercal$, which is exactly the product between a row in Q and two matrices.

This equation can be interpreted as a conic combination, whose parameters are defined by $Q_{b:}$, of the different musical patterns defined by $\mathcal{G}_{(3)}(W \otimes H)^\intercal$.

6.2.2 Reminder of SSAE

A neural network layer, applied to an input x, can be seen as the composition of an affine transformation of x, defined with weights R and a bias b, and a nonlinear activation function σ, such that the output of this layer is equal to $\sigma(Rx + b)$.

In practice here, the activation function σ is set as the ReLU function, except for the latent space [1], where the activation can be the identity function (no nonlinearity) or any differentiable function, with particular examples presented in Section 5.4.5. The activation function for the latent space is denoted as σ_q. AutoEncoders are composed of two parts: an encoder and a decoder.

1. We recall that the use of the ReLU function on the latent space could lead to null latent variables.

In Section 5.4, SSAEs were introduced to compress each barwise time-frequency representation $\mathcal{X}_{(3)_{b:}}$ with an encoder, resulting in a latent representation $Q_{b:}$ (potentially negative) and a decoder which computes an approximate $\widehat{\mathcal{X}_{(3)_{b:}}}$.

In details, the encoder, composed of l layers, results in $Q_{b:} = \sigma_q(R_l^e\, \sigma(R_{l-1}^e\, \sigma(...(R_1^e\, \mathcal{X}_{(3)_{b:}} + b_1^e)...) + b_{l-1}^e) + b_l^e)$, and the decoder computes $\widehat{\mathcal{X}_{(3)_{b:}}} = R_l^d\, \sigma(R_{l-1}^d\, \sigma(...(R_1^d\, Q_{b:} + b_1^d)...) + b_{l-1}^d) + b_l^d$.

6.2.3 AE-NMF

In their work [SV17], Smaragdis and Venkataramani proposed a neural network model which we denote "AE-NMF". An AE-NMF aims at reproducing the structure of NMF in an AutoEncoder, by imposing nonnegativity on the layers. Denoting as $M \approx UV$ the NMF, the AE-NMF is composed of only one hidden layer (which defines the latent space), resulting in a one-layer encoder and a one-layer decoder.

> The latent representations are interpreted as the U matrix, and the weights of the decoder are interpreted as the V matrix.

Hence, the AE-NMF computes a matrix $\hat{M}$ such that $\hat{M} = UV \approx M$. In the original AE-NMF [SV17], the encoder is enforced to be related to the pseudo-inverse of V, but we instead choose not to formally relate the encoder with the NMF framework for subsequent work, in order to relax the optimization paradigm.

In an AE-NMF, the weights V of the decoder are not constrained to be nonnegative; nonnegativity is instead imposed via the (nonnegative) activation function. Enforcing nonnegativity via the activation function extends the subspace of solutions: a nonnegative matrix V can be obtained in the optimization scheme of the AutoEncoder, meaning that the entire subspace of solutions spanned by NMF is included in the subspace of the AE. Still, the AE-NMF can in addition compute a V matrix with some negative coefficients, and, according to the authors, this larger subspace is beneficial to the decomposition. In addition, this formulation is standard in neural networks optimization schemes, and allows for a diversity in the choice of the activation functions.

The original AE-NMF [SV17] uses the softplus activation function, defined as $\text{softplus}(x) = \log(1 + e^x)$, but the ReLU is also a nonnegative activation function. Figure 6.1 presents a schematic architecture of an AE-NMF.

In addition, this "shallow" AE-NMF, composed of only one hidden layer, can be extended into a multilayer AE-NMF, which allows to learn more complex nonlinear mappings between the input and the output of the neural network [SV17], or can be extended to reproduce variants of NMF [VSS17].

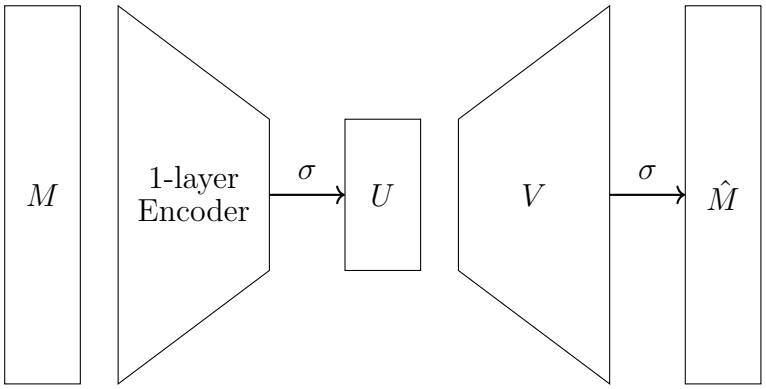


Figure 6.1 – Schematic representation of an AE-NMF.

The network minimizes some loss function $d(M, \hat{M})$ between the input and the approximation, here the KL-divergence with an additional sparsity constraint on the latent representations (penalization via the l_1 norm).

The AE-NMF model is evaluated on the source separation task [SV17]. Experimental results with the one-layer AE-NMF are equivalent or worse than with NMF (depending on the number of factors in the decomposition), but are improved when using the multilayer AE, showcasing the interest for such model.

In addition, one of the main assets of AE-NMF is to recast NMF in the neural network's framework, where it can be easily modified, for instance with constraints or numerous layers, while it may be practically more difficult in the traditional NMF framework.

6.2.4 AE-NTD

We introduce the "AE-NTD" model, extending the previous AE-NMF for the NTD framework.

> The idea of the AE-NTD is to interpret Q, the barwise representation of the song in NTD, as the latent representations, and to use $\mathcal{G}_{(3)}(W \otimes H)^\intercal$ as a two-layer decoder.

This is represented in Equation 6.1. A schematic architecture is presented in Figure 6.2.

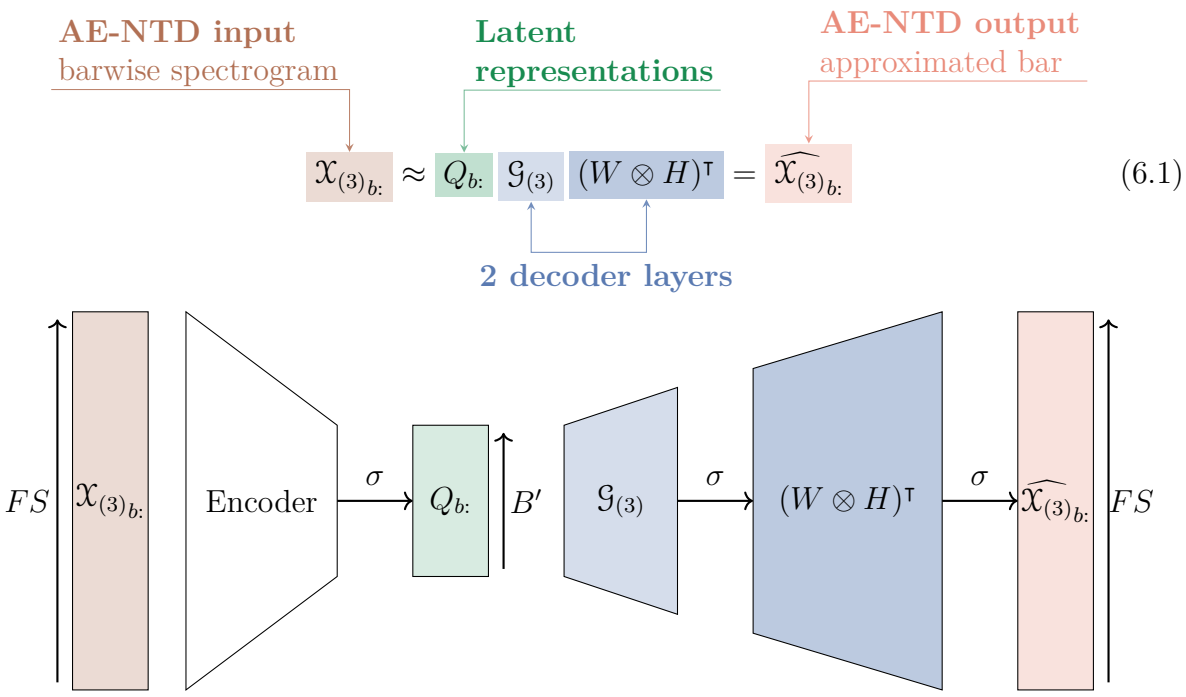


Figure 6.2 – Schematic representation of an AE-NTD. The encoder can be of any shape (fully-connected, convolutional, ...)

Decoder

In an AE-NTD, the decoder is composed of 2 fully-connected layers (without bias), one representing the unfolded $\mathcal{G}_{(3)}$ matrix, and the second one representing the Kronecker product $(W \otimes H)^\intercal$. The structure of the Kronecker product is enforced in the last layer, *i.e.* matrices W and H act as the weights of the fully-connected layers [2].

As for AE-NMF, these weights are not constrained to be nonnegative, and we instead enforce nonnegativity using a nonnegative activation function σ, here the ReLU for consistency with the previously-defined SSAE.

Encoder

The encoder of the AE-NTD can, in theory, be of any shape. Indeed, while it is conventional to keep the same architecture between the encoder and the decoder when designing an AutoEncoder, in principle, both can be different.

In practice, as a proof of concept and following the two-layer NTD-based decoder,

2. The Kronecker product being enforced in the weights of the layer, it may be considered as a particular type of fully-connected layer.

we implement a fully-connected encoder, composed of two fully-connected layers. In particular, this encoder is exactly the encoder of the FC SSAE presented in Appendix B.1, with an additional nonnegative activation function on the latent space, to ensure the nonnegativity of the Q matrix.

While the ReLU is a nonnegative function, it can lead to null latent representations (*i.e.* a bar b represented by $Q_{b:} = 0_:$), which was observed in practice in preliminary experiments. To counteract this side effect, a small constant is added to the activation function, resulting in the activation function $\sigma_q(x) = \text{ReLU}(x) + 10^{-10}$ on the latent space.

In addition, σ_q must be implemented immediately before the latent space. In that sense, and conversely than for the SSAEs and the other encoder layers (here, only one, but many more could be implemented), the activation function for the latent space is implemented **after** the batch normalization layer.

Note that the encoder and the decoder differ in some additional points: firstly, the Kronecker product between W and H in the last layer of the decoder is not implemented in the first layer of the encoder. Hence, the number of parameters of these two layers differ, from $FF' + SS'$ in the last layer of the decoder (sum of the sizes of W and H) to $FF'SS'$ in the first layer of the encoder (size of the result of the Kronecker product).

In addition, a fully-connected layer is composed of weights, represented by a matrix multiplication, and a bias, resulting in an affine transformation of the input, while NTD-based layers are only composed of matrix multiplication, hence a linear transformation.

Finally, the encoder is composed of batch normalization layers, which are not implemented in the decoder.

In that sense, the encoder layers contain more parameters than the decoder layers, while the matrix weights are of the same sizes. The network is presented in Figure 6.3.

Musical Patterns in the AE-NTD

The AE-NTD are developed so as to mix both NTD and SSAE frameworks. Section 4.5.2 studied the musical patterns, computed from the NTD, as the product $P = W\mathcal{G}_{::b}H$.

Still, the formalism in AE-NTD is not exactly the same as for NTD: the nonnegativity is not enforced on the matrices but on the matrix products (with the activation function), and the product is under its matrix form, not its tensor one. Thus, a musical pattern in an AE-NTD is defined as[3] $P = \sigma\left(\sigma(\mathcal{G}_{:b})\,(W \otimes H)^\intercal\right)$, *i.e.* the particular output for each

3. In this equation, P is a vector, but may be refolded into a matrix easily.

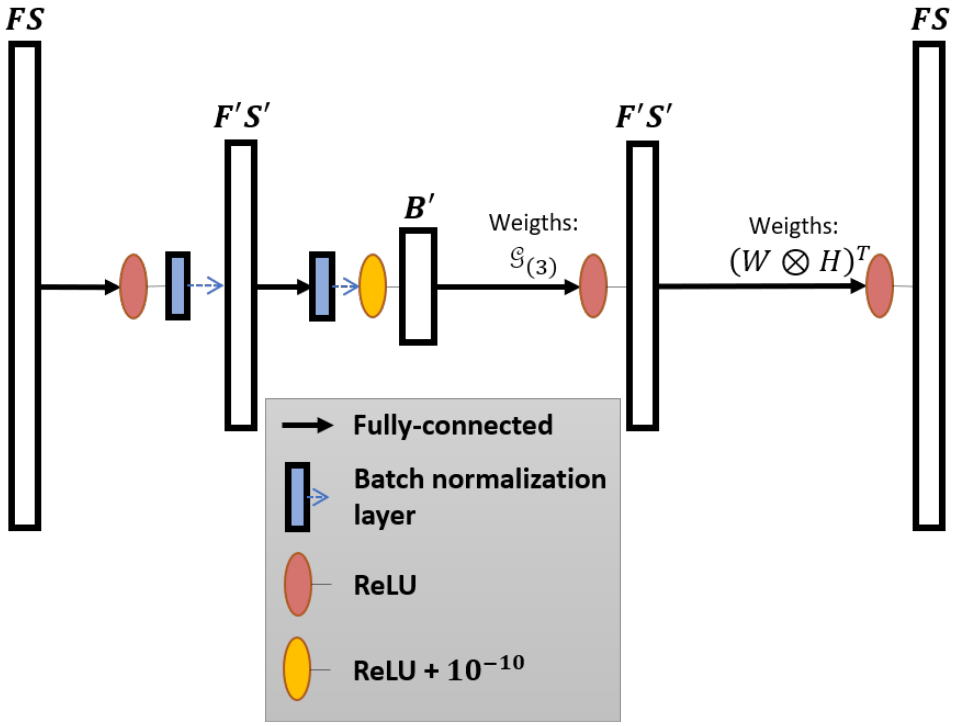


Figure 6.3 – Architecture of the AE-NTD.

dimension of the latent space.

6.3 Motivations

6.3.1 Study the Performance-Interpretability Trade-Off

Both previous Chapters 4 and 5 highlighted a trade-off between the interpretability of the compression outputs and the segmentation performance. In particular, this trade-off is observed between NTD, which computes interpretable patterns in its factorization, and AutoEncoders, which obtain better segmentation results.

By mixing both paradigms, an objective is to study in more depth the performance-interpretability trade-off and, if possible, to obtain a decomposition both interpretable and highly performant.

6.3.2 Increasing the Expressiveness of the Decomposition

The AE-NMF [SV17] was motivated by a greater expressiveness of the AE-NMF paradigm compared to the standard NMF. In particular, this greater expressiveness stems from a larger subspace of solutions in the AE-NMF compared to NMF (due to the non-negative activation function instead of the nonnegativity constraint) and the possibility to increase the depth of the network (*e.g.* the multilinear AE-NMF, which outperforms the one-layer AE-NMF), this latter argument being particularly studied in the recent literature for nonnegative factorization models [DGS21]. Both arguments apply to AE-NTD.

The expressiveness of the decomposition could also benefit from the large neural network literature, for instance by adapting the model to use waveforms as inputs [VTS20], to benefit from transfer learning [WKW16] and/or representation learning [BCV13], or to use new optimization schemes, such as the triplet loss paradigm [SKP15], presented for the SSAE in Section 5.4.5.

Additionally, recent algorithms make use of the neural network formalism in a new optimization scheme, called "unrolling", which broadly consists of implementing traditional optimization methods by means of neural networks, for instance NMF [HLW14], see [MLE21] for a recent overview. Unrolling was already applied to audio signal processing [LHW15].

In this work, the AE-NTD remains a "shallow" neural network model, but, even if these aspects are not treated in the current work, they constitute perspectives.

6.3.3 Technical Benefits

An additional advantage of formalizing NTD in the neural network paradigm is the possibility to benefit from the toolboxes and technical advances of the neural network community. Indeed, high-performance computing is an important aspect of neural network computation, due to the gigantic calculus complexity necessary to train some models, and the standard toolboxes (such as *Pytorch* [Pas+19], used in this thesis for the SSAEs and the AE-NTD) use State-of-the-Art methods to optimize computation.

In addition, the neural network formalism and the associated toolboxes ease the implementation of constraints, such as the activations functions on the latent space presented in Section 5.4.5, which may be harder to implement in the traditional matrix and tensor factorization formalisms.

6.4 Experiments

6.4.1 Technical Details

AE-NTD are developed in the same framework as SSAEs, *i.e.* using *Pytorch* 1.8.0 [Pas+19]. Remaining implementation details can be found in Section 5.4.3. In the current experiments, we only consider the Euclidean distance and the KL-divergence as loss functions, due to the poor performance of the IS-SSAE in Section 5.4. This leads to Euclidean- and KL-AE-NTDs.

As already presented in the motivations, the goal of the AE-NTD paradigm is to mix the interpretability of NTD with, potentially, better segmentation results. In that sense, experiments focus on both the structural segmentation and pattern uncovering tasks, to study both performance and interpretability aspects.

The structural segmentation task focuses on the RWC Pop dataset, and the NNLM spectrograms, this feature being the best-performing nonnegative feature for both NTD and AE. On the other hand, the pattern uncovering task focuses on the STFT of *Come Together* by The Beatles, as in Section 4.5.2.

We recall that the dimensions of NTD must be set prior to the decomposition. Consequently, this is also the case for an AE-NTD. Hence, for the structural segmentation task, dimensions are set to $F' = S' = B' = 24$. Indeed, 24 corresponds both to the dimension of the latent space in the SSAEs experiments, in Section 5.4.4, and to the most frequently picked dimension (considering all matrices) in the cross-validation process for NTD experiments, in Section 4.5 [4]. In the pattern uncovering task, as in Section 4.5.2, dimensions are set to $F' = 32, S' = 12, B' = 10$.

6.4.2 Interpretation of the NTD-based Decoder

The interpretability of the NTD outputs are not guaranteed theoretically speaking, as discussed in Section 4.4.2. The decoder of the AE-NTD mimicking the shape of an NTD decomposition (related to the Q matrix), there is similarly no guaranty that the AE-NTD results in an interpretable decomposition.

For the AE-NMF, the authors argue that enforcing sparsity on the latent space is a sufficient strategy to favor the interpretability of outputs [SV17], which is a common

4. Cross-validation set most than half of matrices to the dimension 24 when studying the NNLM spectrograms.

constraint in NMF [OP14] and NTD [MHA08].

In this work, we rather explore three different initialization strategies for the matrices W, H and $\mathcal{G}_{(3)}$ in the decoder. The rationale of experimenting different initializations is to study the impact of prior information on the final representation, and see if the trade-off between interpretability and segmentation results appearing in the conclusions of Chapter 5 is present in the AE-NTD.

These initializations only impact the decoder, and the encoder is always initialized at random, following previous work on SSAEs (Section 5.4). The three initializations are listed below:

— "Random Init" - Random decoder: the least informed initialization technique, which can be considered as a baseline for comparison. In this scenario, the decoder is fully initialized at random, following the "kaiming" distribution, as for the SSAE and the encoder of the AE-NTD.

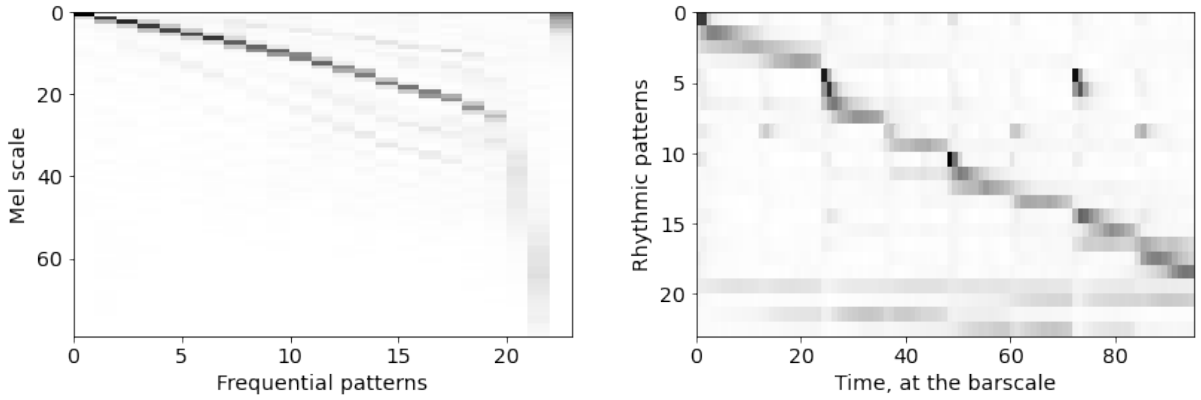
— "General Init" - initializing W and H with generic matrices: in this condition, both matrices W and H are initialized following prior knowledge, and $\mathcal{G}_{(3)}$ is initialized at random. More precisely, W and H are initialized as dictionaries, based on an average of the NTD computed on the RWC Pop dataset.
Indeed, computing the NTD on the one hundred songs of the RWC Pop dataset (with dimensions F', S' and B') results in one hundred W^{NTD} and H^{NTD} matrices. Then, two large matrices $W^{NTD}_{RWC} \in \mathbb{R}_+^{F \times 100F'}$ and $H^{NTD}_{RWC} \in \mathbb{R}_+^{S \times 100S'}$ are obtained by concatenating all the columns of these one hundred matrices, *i.e.* two matrices containing all the frequential and rhythmic templates obtained for all the songs.
Finally, we apply the k-means algorithm [Mac67] on both of these matrices, resulting in respectively F' and S' cluster centroids of frequential and rhythmic templates, leading to matrices $W_{RWC} \in \mathbb{R}_+^{F \times F'}$ and $H_{RWC} \in \mathbb{R}_+^{S \times S'}$. These matrices then serve as initialization for the matrices in the decoder in the "General Init" paradigm.

— "NTD Init" - initializing the decoder with the NTD: in this condition, all matrices (W, H and $\mathcal{G}_{(3)}$) are initialized as the output of the NTD performed on this song. In that sense, the AE-NTD tries both to learn the Q matrix of NTD as its latent representations, and to enhance the W, H and $\mathcal{G}_{(3)}$ matrices in order to obtain better estimates.
In this condition, the parameters of the batch normalization on the latent space are also initialized as the average and standard deviation of the Q matrix obtained

from NTD [5].

These three conditions constitute a gradation of the level of prior knowledge enforcement, via NTD outputs. This gradation starts with the NTD Init (where the network is exactly initialized with the output of an NTD) and ends at the Random Init condition (where NTD is only enforced by structure). The General Init condition is here an intermediate condition (no particular NTD is used, but it still uses a general centroid of all the decompositions), and many more intermediate conditions could be investigated, in particular using supervision and learning schemes.

Matrices W_{RWC} and H_{RWC} for the Euclidean-AE-NTD are shown in Figure 6.4, based on the NNLM spectrograms and NTD dimensions $F' = S' = B' = 24$. For visualization purposes, columns of these matrices are reordered, highlighting a staircase shape.



(a) W_{RWC} (b) $H^{\intercal}_{RWC}$

Figure 6.4 – General matrices W_{RWC} and H_{RWC} computed from the Euclidean-NTD of the NNLM spectrograms of the whole RWC Pop dataset.

Indeed, in the W_{RWC} matrix, the activations in each column (except for the last 3) seem to focus on a particular row, which we interpret as the fundamental frequency of a note (in the Mel scale), and the different columns nearly span the lowest-pitched fundamental frequencies in a sequential order. The emphasis on the lowest-pitched notes may be explained by the power discrepancies between low- and high-pitched notes, this phenomenon being already presented in Section 2.2.4.

In addition, numerous columns of W_{RWC} exhibit a comb aspect, *i.e.* stronger activations for frequencies which are multiples of the fundamental frequency, in the Mel scale.

5. We observed that this was important to learn the Q matrix as latent representations, in particular for the KL-AE-NTD.

This is consistent with previous work on NMF-based decompositions of music audio signals (as the "harmonicity constraints" in [VBB09]), and follows the physical interpretation of musical notes, consisting of a fundamental frequency and harmonic partials.

In the H_{RWC} matrix, a similar staircase shape arises according to the time indexes. It seems that the columns of H_{RWC} (rows of $H_{RWC}^{\intercal}$ in the visualization) focus on particular beats of the bar, sometimes one per column, hence activating a precise time instance in the bar, and sometimes several time instances in the bar (in general located on fourth or eight notes), as a rhythmic pattern. The last columns of both W_{RWC} and H_{RWC} span large zones, which could constitute residual noise or centroids of inaccurate and/or scarce estimations.

In the musical barwise interpretation of NTD, presented in Section 4.4, W and H act as dictionary matrices, *i.e.* general matrices of frequential and rhythmic templates which can be mixed in $\mathcal{G}$ to result in barwise patterns. In that regard, W_{RWC} and H_{RWC} seem appropriate.

We expect the structural segmentation scores to decrease with the gradation, *i.e.* to obtain the largest segmentation scores in the Random Init condition, and the lowest ones in the NTD Init condition, and even with the NTD itself. Conversely, we expect higher SDR in the pattern uncovering task for the NTD and the NTD Init AE-NTD, and low SDR for in the Random Init. The General Init should represent an intermediate condition in both experiments, supposing that W_{RWC} and H_{RWC} matrices are relevant.

Hence, experiments focus on answering **Questions 13** and **14**.

Question 13 *How does prior knowledge (i.e. the initialization of the decoder) impact the segmentation scores?*

Question 14 *How does prior knowledge (i.e. the initialization of the decoder) impact the quality of the decomposition?*

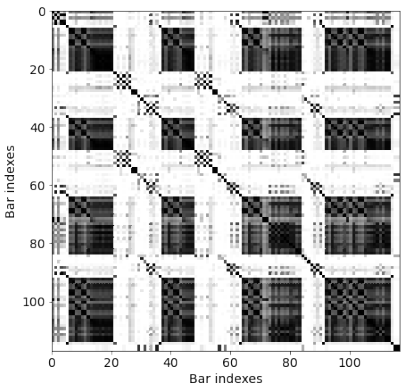
6.4.3 Structural Segmentation

As for the NTD and SSAE, the structural segmentation task is studied on autosimilarities of the Q matrix, with the CBM algorithm. These experiments are computed on the RWC Pop dataset only, leading to average scores on the dataset for the different metrics.

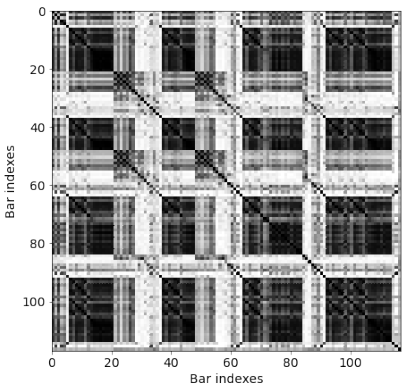
As in Section 5.4, AE-NTD are computed with five different seeds, and results are aggregated in boxplots. The scores displayed in the boxplots represent the median of the five average scores.

Euclidean-AE-NTD

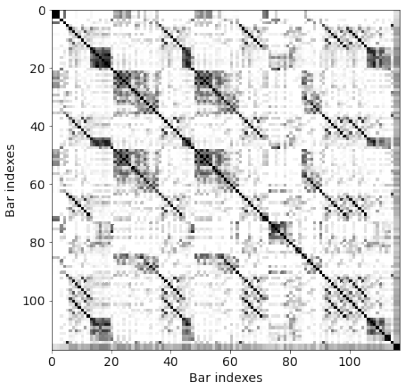
Firstly, we focus on the Euclidean-AE-NTD. Cosine autosimilarities of the song *POP01*, in the different conditions, are presented in Figure 6.5.



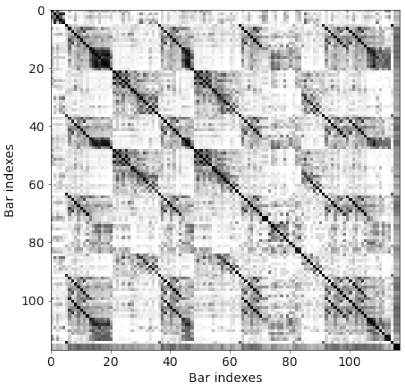
(a) Euclidean-NTD



(b) AE-NTD, NTD Init



(c) AE-NTD, General Init



(d) AE-NTD, Random Init

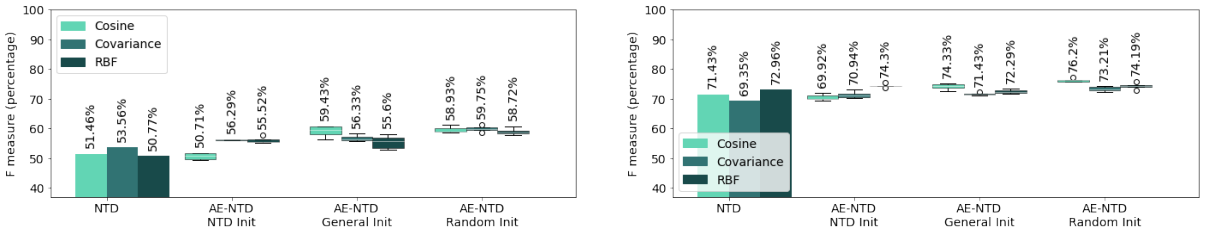
Figure 6.5 – Cosine autosimilarities for the Euclidean-NTD and Euclidean-AE-NTD, computed on the NNLMS of the song *POP01*, for $F' = S' = B' = 24$.

Figure 6.6 presents the segmentation results for the NTD and the AE-NTD, depending on the similarity function. Except for the NTD Init, the Cosine autosimilarity obtains better or similar scores than the other functions, suggesting that the compression mechanism leads to a better notion of dissimilarity, as discussed in previous chapters.

In the NTD Init condition, the RBF autosimilarity is better performing than both others (in particular Cosine), which may indicate a worse dissimilarity notion compared to the other compression schemes. Empirically, the setback of the NTD Init condition compared to the others is observed in Figure 6.5, with a visually less obvious block structure in the NTD Init condition compared to the NTD, and an overall lower contrast (*i.e.*

larger similarity overall) compared to the General and Random Init. Hence, the NTD Init condition seems maladapted to enhance the similarity and dissimilarity between bars, *i.e.* maladapted to highlight the structure in the autosimilarity.

Conversely, the General Init and Random Init improve segmentation performance, and, empirically, exhibit a better contrast between zones of high and low similarity in Figure 6.5. Still, in this particular example, the block structure seems visually less obvious for the AE-NTD than for the NTD.



(a) $F_{0.5}$ (b) F_3

Figure 6.6 – Segmentation results for the Euclidean AE-NTD, depending on the initialization of the decoder, compared with the NTD.

In addition, Table 6.1 presents the reconstruction errors according to the different conditions, averaged on the RWC Pop (in particular, avg(errors) $\pm$ std(errors)) [6]. The reconstruction error is computed, for each song, as the average of the elementwise error between the original tensor and the reconstructed one, here with the Euclidean distance, but similarly for the KL-divergence in subsequent results [7]. Precisely, denoting as $\mathcal{X}$ and $\hat{\mathcal{X}} \in \mathbb{R}^{F \times S \times B}$ respectively the input and output of the AE-NTD, the reconstruction error e is computed as $e = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_2^2}{FSB}$.

The reconstruction error provides an additional conclusion: adding prior knowledge results in better estimates, but the quality of the estimate is not correlated with the segmentation score. Indeed, the largest reconstruction error is obtained with the Random Init decoder, which is also the best-performing AE-NTD segmentation-wise.

In addition, the initialization directly impacts the quality of the estimate, and, as suggested for the AE-NMF [SV17], the neural network paradigm can lead to better estimates, as presented here for both General Init and NTD Init, which outperforms the NTD in

6. Note that we use the same notation as for the median plus or minus the MAD for the different runs of the SSAEs and AE-NTDs. We apologize for the clash in notation.

7. In general, reconstruction errors are divided by the norm of the tensor. This is not the case here, for implementation issues: the norm of the tensor used for normalization depends on the loss function, which could equivalently be the Euclidean distance or a β-divergence.

terms of reconstruction error.

Method		Reconstruction error
NTD		8.06
AE-NTD	NTD Init	**4.03**±0.02
	General Init	5.68±0.02
	Random Init	22.69±0.60

Table 6.1 – Average elementwise reconstruction error, in term of Euclidean distance, for the different techniques computed on the RWC Pop dataset.

KL-AE-NTD

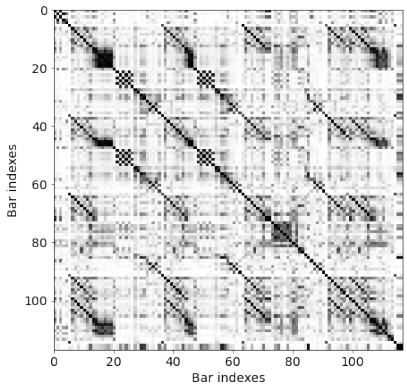
We pursue the experiments with the KL-AE-NTD. Examples of Cosine autosimilarities are presented in Figure 6.7. Figure 6.8 presents the segmentation results of the KL-AE-NTD, comparing the NTD and the different initialization conditions, for all the similarity functions. Table 6.2 presents the average elementwise KL-divergences between the original tensors and the estimates, *i.e.* $\frac{d_\beta(\mathcal{X}-\hat{\mathcal{X}})}{FSB}$.

Method		Reconstruction error
NTD		0.41
AE-NTD	NTD Init	**0.36**±0.003
	General Init	0.90±0.011
	Random Init	1.04±0.128

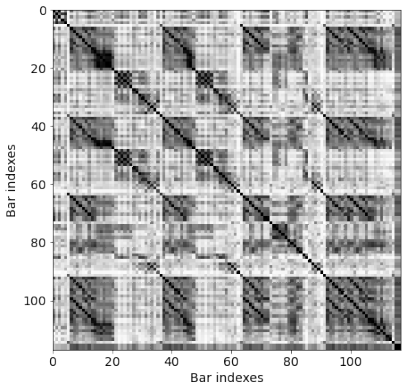
Table 6.2 – Average elementwise reconstruction error, in term of KL-divergence, for the different intializations.

The NTD Init obtain similar (or slightly worse) segmentation scores than the NTD, and, as for the Euclidean-AE-NTD, the best performance are obtained with the RBF autosimilarity. Hence, the global trend for both loss functions is that the NTD Init condition does not improve the performance in terms of structural segmentation compared to those of the NTD, while the reconstruction errors are lower (*i.e.* it results in better estimates in a data viewpoint). This conclusion probably stems from the CBM algorithm, with a worst contrast and/or block structure in the NTD Init condition.

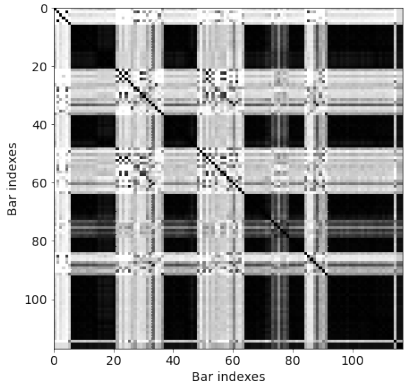
The Random Init outperforms all other conditions, which is consistent with the results of the Euclidean-AE-NTD. Unexpectedly, the General Init condition performs worse than the others, with all similarity functions, while it was expected to obtain (at least) similar



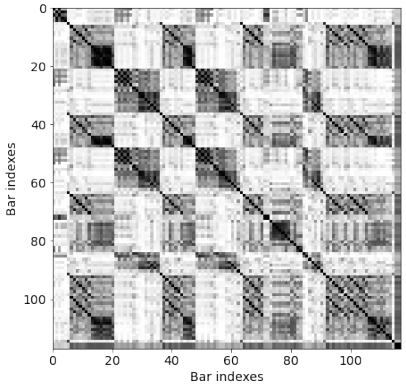
(a) KL NTD



(b) AE-NTD, NTD Init



(c) AE-NTD, General Init



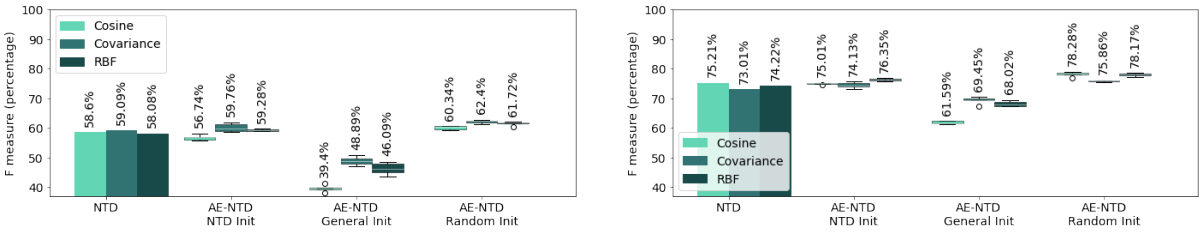
(d) AE-NTD, Random Init

Figure 6.7 – Cosine autosimilarities for the KL-NTD and KL-AE-NTD, computed on the NNLMS of the song *POP01*, for $F' = S' = B' = 24$.

performance than those of NTD. We study this condition specifically in Appendix C, with the goal to explain this drop in performance.

Overall, and except for the General Init in the KL-AE-NTD, the gradation of the level of prior knowledge enforcement is correlated to the segmentation performance, and, the less prior knowledge is enforced, the better are segmentation performance.

Additionally, when comparing the KL- and Euclidean-AE-NTD, both NTD Init and Random Init conditions react similarly to the choice of the similarity function in terms of segmentation performance. The reconstruction errors are also acting similarly (*i.e.* a better reconstruction error for the NTD Init condition compared to the Random Init).



(a) $F_{0.5}$ (b) F_3

Figure 6.8 – Segmentation results for the KL AE-NTD, depending on the initialization of the decoder, compared with the NTD.

6.4.4 Pattern Uncovering

The interpretability of the AE-NTD outputs is evaluated on the pattern uncovering task, as defined in Section 4.5.2. The goal of this section is to study the objective audio quality of the output of a decomposition in terms of SDR (Signal-to-Distortion Ratio), comparing two audio signals in energy ratios. The current experiments follow those already presented in Section 4.5.2, but we reintroduce the important notions in what follows.

Reminder of the Task

Song Scale and Pattern Scale SDR are estimated at two scales, namely the song scale and the pattern scale. The song scale studies the reconstruction of the whole song, *i.e.* compares the signal computed for the whole song by the AE-NTD with the original signal of the song. This results in a unique SDR for the entire song.

The pattern scale studies the quality of all patterns, *i.e.* the products $W\mathcal{G}_{::b'}H$ for NTD and $\mathrm{ReLU}(\mathrm{ReLU}(\mathcal{G}_{:b})\,(W \otimes H)^\intercal)$ for the AE-NTD. Here, the NTD and AE-NTD compute $B' = 10$ patterns, hence resulting in ten SDR values. In particular, a pattern being a barwise spectrogram, it must be studied according to a unique bar in the original song, namely the **associated bar**. In these experiments, the associated bar is chosen similarly than for NTD, *i.e.* the associated bar for the pattern b' is the bar b of maximal coefficient $Q_{b,b'}$ in matrix Q.

Results are shown as the average of these ten SDR scores plus or minus the standard deviation (*i.e.* $\mathrm{avg(SDR)} \pm \mathrm{std(SDR)}$) [8].

8. Note that we use the same notation as for the median plus or minus the MAD for the different runs of the SSAEs and AE-NTDs. We apologize for the clash in notation.

Phase Estimation The song is reconstructed as $\hat{X}$, and patterns are obtained from the AE-NTD as the product $P = \mathrm{ReLU}(\mathrm{ReLU}(\mathcal{G}_{:b})\,(W \otimes H)^\intercal)$, both being real-valued spectrograms.

Still, our experiments are designed so as to evaluate signals. An audio signal can be computed from a complex-valued STFT spectrogram by applying the Inverse STFT. Thus, starting from a real-valued spectrogram ($\hat{X}$ or P), corresponding to the modulus of STFT coefficients, we additionally need to estimate the argument of each of these STFT coefficients. Arguments of STFT coefficients correspond to the "phase information".

Three methods are employed, depending on the scale of study. At the song scale, we either use the Griffin-Lim algorithm [GL84] or use directly the original phase information. At the pattern scale, we either use the Griffin-Lim algorithm or softmasking. All these techniques are detailed in Section 4.5.2.

Results

Tables 6.3 and 6.4 present the SDR respectively for the Euclidean- and KL-based methods. Results show that, in most conditions, increasing the level of prior knowledge enforcement also increases the SDR, a few exceptions being obtained in the General Init condition. Notably, the NTD Init condition generally obtains better results than the Random Init condition for both loss functions, and in the few exceptions, performance are similar.

When comparing the NTD performance with those of the AE-NTDs, one has to discriminate the conclusions according to the loss function: the NTD obtains the best performance in every condition with the KL-divergence, while, with the Euclidean distance as loss function, performance of the NTD are outperformed in three among the four conditions.

The general trend for the Euclidean loss function is that the AE-NTD in the NTD Init condition obtains similar or better results than the NTD (especially considering the standard deviations at the pattern scale), while the General and Random Init conditions obtain similar or worse performance than the NTD.

Overall, these results conclude towards an advantage of using prior knowledge for the interpretability of patterns, in particular when comparing the NTD Init condition with the Random Init condition.

Method & Phase Retrieval		Song scale		Pattern scale	
		Griffin-Lim	Original Phase	Griffin-Lim	Softmasking
NTD		-38.47	4.35	-20.81 ± 2.44	**16.71** ± 5.10
AE-NTD	NTD Init	-38.70	**5.05**	**-17.95** ± 2.58	12.85 ± 3.85
	General Init	**-37.42**	0.99	-19.90 ± 3.72	7.59 ± 3.53
	Random Init	-40.63	0.59	-21.44 ± 4.59	8.27 ± 2.70

Table 6.3 – SDR results, both at the song scale and averaged for the different patterns, for the Euclidean-NTD and Euclidean-AE-NTD.

Method & Phase Retrieval		Song scale		Pattern scale	
		Griffin-Lim	Original Phase	Griffin-Lim	Softmasking
NTD		**-34.53**	**6.08**	**-17.69** ± 3.03	**25.94** ± 5.48
AE-NTD	NTD Init	-36.46	5.02	-19.21 ± 3.03	13.69 ± 4.35
	General Init	-40.17	-6.72	-21.81 ± 0.83	2.71 ± 1.09
	Random Init	-38.91	2.46	-19.47 ± 3.51	14.90 ± 18.18

Table 6.4 – SDR results, both at the song scale and averaged for the different patterns, for the KL-NTD and KL-AE-NTD.

6.4.5 Experimental Conclusions

In light of these results, two experimental conclusions can be drawn:

Experimental conclusion 13 *On the RWC Pop dataset, enforcing prior knowledge related to the NTD negatively impacts the segmentation scores.*

Experimental conclusion 14 *On the RWC Pop dataset, enforcing prior knowledge related to the NTD positively impacts the quality of the decomposition, both in terms of reconstruction error and interpretability of the patterns in the decoder.*

Hence, there is indeed an experimental trade-off between segmentation scores and interpretability of the output. Except for the General Init KL-AE-NTD, increasing the level of prior knowledge enforcement negatively impacts the segmentation scores and positively impacts the quality of the musical patterns, *i.e.* the musical interpretation of the outputs.

Finally, the AE-NTDs are designed following the SSAEs. Hence, it would be informative to compare the performance of both paradigms. Intuitively, the AE-NTD is much more constrained than the SSAEs: nonnegativity is implemented on the latent space, and the decoder is constrained to follow an NTD structure. In addition, the FC SSAE

presented in Appendix B.1, which is close to the current AE-NTD (notably in their decoders architectures), is largely worse performing than the Conv SSAE, whose results are presented hereafter. Thus, one can expect a drop in performance when using AE-NTD compared to using SSAEs.

Table 6.5 compares the segmentation results of the best AE-NTD (*i.e.* in the Random Init condition) with the segmentation results of the SSAEs. Results show that the performance of the AE-NTD are worse or similar than the performance of the SSAEs. Still, results with the KL-divergence are similar, and the largest difference is for the $F_{0.5}$ of Euclidean networks.

While the best technique is the SSAE, as expected, the performance are really close, which was not expected [9].

Compression method		$F_{0.5}$	F_3
Euclidean	Conv SSAE	**62.61%** ±1.23%	**76.47%** ±1.66%
	AE-NTD, Random Init	58.93% ±0.28%	76.20% ±0.70%
KL	Conv SSAE	**60.95%** ±1.48%	77.18% ±1.06%
	AE-NTD, Random Init	60.34% ±0.49%	**78.28%** ±0.55%

Table 6.5 – Comparison of the segmentation scores of the Conv SSAEs and the Random Init AE-NTDs, for the NNLMS and the RWC Pop dataset.

6.5 Conclusions

This chapter mixed both SSAE and NTD paradigms in a larger framework, called AE-NTD, by reinterpreting the factors of NTD into a latent representation and a decoder for an AutoEncoder.

The current paradigm must be considered as a first attempt to mix NTD with AutoEncoders, thus this paradigm should be extended in future work, some perspectives being listed in the motivations (see Section 6.3). In particular, in their seminal paper [SV17],

9. In addition, it can be seen by comparing the results presented here with the results presented for the FC SSAE in Appendix B.1 that the use of an NTD decoder is largely beneficial when using a fully-connected encoder. This could be explained by the fact the subspace of solution is narrowed by the structure in the decoder, hence refining the optimization scheme.

Smaragdis and Venkataramani finds that the one-layer AE-NMF obtains worst or similar performance than the standard NMF, while the multilayer AE-MF outperforms the standard NMF. Hence, increasing the depth of the AE-NTD could be a promising future direction.

Nonetheless, some first conclusions can be drawn from the current experiments. In its current version, the AE-NTD does not achieve the optimistic objective of combining both very high segmentation performance and an interpretable representation of the song. Nonetheless, the experiments presented in this chapter confirm and study in more depth the trade-off between segmentation performance and interpretable representations.

It also seems that the KL-divergence is less an asset for neural networks than for NTD, as already sketched in Chapter 5.

In addition, the NTD Init condition does not seem to improve the NTD decomposition itself, and the Random Init condition does not outperform the SSAEs performance. The General Init condition is an interesting intermediate condition with the Euclidean loss function, but does not achieve sufficient performance with the KL-divergence.

In that sense, it may be interesting to study different initialization conditions, more prone to take advantage of both models, and to extend the current AE-NTD formalism.

Chapter 7

Conclusion - Outro

This thesis has studied the Music Structure Analysis task, consisting of representing a song in sections (such as "chorus", "verse", "solo" etc), and more specifically the subtask of **structural segmentation**, consisting of retrieving the boundaries between sections within a music piece. In addition, some of the presented techniques were used in a secondary task of pattern uncovering, consisting of extracting elementary patterns in music at the barscale.

Three aspects have been covered in this thesis: a barwise processing of music, the CBM algorithm for segmentation, and the use of compression methods to obtain efficient descriptions of music content for the task of structural segmentation.

1. Barwise processing of music. In this thesis, we assumed that the barscale is a relevant scale to study structural segmentation, in particular for Pop music. Some experimental results, comparing beatwise and barwise alignments of boundaries estimated with the State-of-the-Art algorithms, supported this hypothesis. More generally, we may conjecture that a priori aligning the estimation of the existing structural segmentation algorithms on bars could be beneficial.

   Barwise processing was also intended to study redundancies and musical motifs developed at the barscale, and to use these redundancies for the estimation of structure. The experimental results presented throughout this thesis suggest that this viewpoint is relevant and beneficial to the estimation of structure in Pop music.

   This thesis has introduced two representations for barwise processing of music, namely the Barwise TF matrix and the TFB tensor. In particular, the TFB tensor allows to study music according to two different time dimensions, and therefore to model redundancies both within and across bars.

2. The "CBM" algorithm (standing for *Convolutive Block Matching*) is based on the computation of an autosimilarity matrix, representing similarities between pairs of time instances in the song (in our scale, between pair of bars). In line with previous

works in the literature, this algorithm is a dynamic programming algorithm.

The main contribution related to the CBM algorithm is the definition of the score function applying on each segment. The score function focuses on a local homogeneity criterion and a regularity criterion based on the expected segment distribution. This results in an algorithm which turns out to be competitive with the unsupervised State-of-the-Art algorithms for structural segmentation.

Still, the CBM algorithm has not been tested on autosimilarity matrices aligned on other time scales than the barwise one (for instance, the beats), while it could be informative regarding the advantages of using barwise-aligned features. Nonetheless, we conjecture that barwise-aligned features are particularly favorable for the homogeneity criterion, as motifs are more prone to appear at the barscale than at the beatscale.

As a point of particular importance, we noticed that increasing the value differences (the **contrast**) between zones of high and low similarity in the autosimilarity matrix (*i.e.* homogeneous *vs.* dissimilar parts of the songs) largely enhances the performance of the CBM algorithm. We assumed that this gain in performance stems from the focus on the homogeneity criterion in the design of the CBM algorithm. Indeed, increasing the contrast between zones of high and low similarity disambiguates the boundaries between blocks of high similarity, *i.e.* segments.

In that respect, this thesis considered three different similarity functions for the computation of autosimilarity matrices, leading to three different viewpoints on how to consider barwise similarity, thus impacting the contrast. As a consequence, we observed that the choice of the similarity function largely impacts performance when considering the feature representation of songs. The Radial Basis Function appeared to be particularly relevant in that context, compared to the more standard Cosine similarity function.

3. The use of compression schemes on barwise representations of the song. Central to this thesis is the study of the use of several compression schemes on the barwise representations of music, as input to the CBM algorithm. Compression schemes are motivated as alternative strategies for increasing the contrast in autosimilarity matrices. Indeed, compressing the different bars in the song focuses the representation on a few essential attributes, accounting for barwise redundancies and, incidentally, enhancing the dissimilarity between the different bars compared to that in the primary feature space.

In particular, the Nonnegative Tucker Decomposition (NTD), a tensor factorization scheme, has been investigated as one of the compression techniques. Applied to music, NTD can be viewed as a paradigm representing a song with musical patterns, which can then serve as features for the similarity computation. In particular, these musical patterns appear to be interpretable musically-speaking, which may open the use of NTD for tasks such as music generation, recomposition, or musicological analysis in general.

NTD was compared with standard matrix compression schemes, namely Principal Component Analysis (PCA), Nonnegative Matrix Factorization (NMF) and AutoEncoders (AE). In particular, the AutoEncoders presented in this thesis are unsupervised compression schemes at the song scale.

The comparison between the different compression schemes reveals a trade-off between the interpretability of the compressed representations and the segmentation performance. Indeed, while NTD is prone to interpretations as musical patterns, PCA and (unconstrained) AutoEncoders provide better segmentation results. In light of this conclusion, a mix between both NTD and AutoEncoders paradigms, coined "AE-NTD", has been investigated. We believe that AE-NTD represents an original work mixing NTD and AutoEncoders, largely inspired from previous developments mixing AutoEncoders and matrix factorizations. Bridging the gap between low-rank factorizations and AutoEncoders could pave the way for interpretable representations in neural network schemes.

Overall, the compression methods that we investigated obtain competitive results with those of the State-of-the-Art algorithms on the structural segmentation task, showcasing their relevance for uncovering the structure of music. In particular, the performance of compressed representations is less sensible to the choice of similarity functions, compared to the representations in the primary feature space.

Summing up, two alternative strategies have been employed in this thesis for enhancing the notions of similarity and dissimilarity in the feature representation of a song: changing the similarity function and using compression methods. Both strategies work to some extent, but their combination remains to be optimized.

Perspectives: this work calls for further research tracks.

Firstly, the compression schemes are definitely worth devoting additional research. In particular, a clear improvement for these techniques would be to design a criterion to

determine accurate dimensions of the compression model prior to the decomposition. In addition, constraints (typically, sparsity) and supervision or semi-supervision (*e.g.* supervision for W and H only in the NTD) could improve the compression outputs. While the "interpretable" techniques (*i.e.* NTD and AE-NTD, potentially NMF) have provided an interesting result on a case study, further work is needed to consolidate this observation on a more extended set of data. In particular, this interpretable aspect could be showcased on different applications (for instance music generation or recomposition).

Secondly, the CBM algorithm has the potential to be further extended, for instance by studying new kernels, but also by adding mechanisms able to account for the repetition criterion and even further combining different criteria. In particular, we believe that the use of kernels such as the ones developed by Shiu *et al.* [SJK06] could be worth studying.

Thirdly, experimental conclusions should be validated on other datasets, and notably on other musical styles than Pop music. The study of the impact of barwise processing in music deserves to be deepened for other styles of music, especially when the bar division may be ambiguous or rapidly-changing.

To conclude, this thesis must be view as an attempt to explore the structural segmentation of music. However, music structure is still subject to debate in the communities dealing with music analysis (Computational Music, MIR and Musicology). This unsettled situation still results in multiple incomplete viewpoints on music structure, which adds an additional level of complexity in interpreting the results of our models and algorithms.

Citing the song *Lateralus* of the band TOOL, we are (to a certain extent) in the situation where we are "look[ing] through to these infinite possibilities" of studying the musical object, which could be a never ending quest. Still, let us conjecture that part of the beauty in music lies in our ability to experience it without fully grasping it, as, citing *Lateralus* again, "over thinking, over analyzing, separates the body from the mind".

List of Figures

1 Un exemple schématique et idéalisé de matrice d'autosimilarite, extrait de [PMK10]. 7

2 Traitement par mesures de la musique, en tenseur et en matrice. Les dimensions F, S, B sont représentées afin de faciliter la compréhension du processus. 8

3 Décomposition nonnégative de Tucker (NTD) du tenseur construit à partir de la collection de spectrogramme par mesures. La NTD résulte en trois matrices de facteurs W, H, Q, et un tenseur $\mathcal{G}$ (appelé "cœur"). 10

4 Articulations de la thèse. Cette thèse présente les étapes de calcul de la collection de spectrogrammes en mesures, l'algorithme CBM et les différentes méthodes de compression. *Les étapes de représentation en descripteurs et d'estimation des mesures sont issues de travaux extérieurs et, bien qu'importantes, n'ont pas fait l'objet d'un travail de recherche dans cette thèse.* Trois logiciels libres ont été développés durant ce doctorat, chacun étant dédiée à une partie du travail : l'algorithme CBM est inclus dans la librairie *as_seg* [MCB22a], les algorithmes de résolution de la NTD et de la NMF sont inclus dans la librairie *nn_fac* [MC20], et l'ensemble du code informatique associé aux représentations compressées est rassemblé dans la librairie *BarMusComp* [MCB22b]. 11

1.1 Barwise processing of music, resulting in a matrix (the Barwise TF matrix) and in a tensor (the TFB tensor), depending on the technique which is to be used. Dimensions F, S, B are represented so as to ease the understanding of the process, but are not at the same scale in the different representations. 23

1.2 Nonnegative Tucker Decomposition (NTD) of the TFB tensor, *i.e.* the tensor composed of barwise spectrograms. NTD results in three factor matrices W, H, Q, and a "core" tensor $\mathcal{G}$. 25

1.3 This thesis covers the computation of barwise spectrograms, the CBM algorithm for estimating structural boundaries, and the impact of using compressed representations on the estimation quality. *The stages of feature representation and bar estimation, while important to the process, use earlier work from the MIR community, and do not constitute novel contributions in this thesis.* Three open-source toolboxes were developed during this PhD, each dedicated to a part of the work: the CBM algorithm is included in the *as_seg* toolbox [MCB22a], the NTD and NMF resolution algorithms are included in the *nn_fac* toolbox [MC20], and the entire work related to the compressed representations is pooled in the *BarMusComp* toolbox [MCB22b]. 26

2.1 An example of audio signal: passage from the song *Rosetta Stoned* of the band TOOL. 30
2.2 Musical score example 32
2.3 Note values, from top to bottom: a whole note, half notes, quarter notes, eighth notes, sixteenth notes and thirty-second notes. Each line represents the same amount of time. (From Wikimedia, Public Domain.) 35
2.4 Sampling operation ((a), upper) and quantizing operation ((b), lower), extracted from [Mül15]. 38
2.5 Example of STFT magnitude spectrogram, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1). The left figure presents the average magnitude values according to the frequency bins. The right figure presents the log-magnitude spectrogram, *i.e.* the logarithm of the magnitude values of the STFT. 40
2.6 Illustrative example of a Mel filter bank, with only 12 Mel filters, presented on the frequency scale. STFT coefficients are aggregated along the filters. . 41
2.7 Example of Mel, Log Mel and NNLM spectrograms, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1). 43
2.8 Example of chromagram, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1). 43
2.9 Example of MFCC, on a passage from the song *Rosetta Stoned* of the band TOOL (presented in Figure 2.1). 44
2.10 A schematic example of musical structure 45
2.11 An idealized autosimilarity matrix, extracted from [PMK10]. 47
2.12 Representation of the Foote's novelty kernel. 48

2.13 Kernel from Shiu *et al.* [SJK06], designed to find stripes (and so, patterns) repeating every two frames. The parameter n is the number of nonzero elements (18 in this example). 51
2.14 Segmentation results of State-of-the-art algorithms on the RWC Pop and the SALAMI datasets, for beatwise (original) and barwise aligned boundaries. 60
2.15 Barwise sampled spectrogram, *i.e.* a representation where each bar contains the same number S of time frames. 61
2.16 Barwise sampling: zoom on a bar. Here, the bar corresponding to the original feature representation is composed of $\Omega_B = 21$ frames, and the barwise sampled bar of $S = 6$ frames. 62
2.17 Barwise TF matrix. 62
2.18 Distribution of segments sizes, in terms of number of bars, in the annotations. The SALAMI dataset is restricted to the test subset, as detailed in Appendix A.2, and both sets of annotations are mixed here. 64

3.1 Cosine, Covariance and RBF autosimilarities on the song *POP01* of RWC Pop, in the Log Mel feature. 69
3.2 Longest path resolution for a DAG with 4 bars. The lengths of each edge corresponds to the score of the segment between the associated bars, *i.e.* $\gamma([\![b_i, b_j[\![)$. 73
3.3 Autosimilarity restricted to the segment $[\![40, 60[\![$. 76
3.4 Full kernel of size 10 78
3.5 Band kernel, of size 10 78
3.6 Distribution of segments sizes in terms of number of bars, in the annotations. 80
3.7 Distribution of the segment sizes, with the full kernel, according to the autosimilarity matrix. Results are computed on the RWC Pop dataset. . . 83
3.8 Comparison of the F measures, according to the full and band kernels (with different number of bands). Results are computed on the RWC Pop dataset. 84
3.9 Comparison of the F measures, according to the full and band kernels (with different number of bands). Results are computed on the SALAMI dataset. 84
3.10 Distribution of the segment sizes, with the RBF autosimilarity matrix, according to different kernels. Results are computed on the RWC Pop dataset. 85
3.11 Distribution of the segment sizes, with the RBF autosimilarity matrix, according to different kernels. Results are computed on the SALAMI dataset. 86

3.12 Segmentation results depending on the choice of the feature description and on the autosimilarity matrix. Results are computed on the RWC Pop dataset, with the 7-bands kernel and the modulo 8 penalty function. 88
3.13 Segmentation results depending on the choice of the feature description and on the autosimilarity matrix. Results are computed on the SALAMI dataset, with the 15-bands kernel and the modulo 8 penalty function. . . . 88
3.14 Segmentation results of the best CBM algorithm, compared to the other State-of-the-Art algorithms. Results are computed on the RWC Pop dataset. 89
3.15 Segmentation results of the best CBM algorithm, compared to the other State-of-the-Art algorithms. Results are computed on the SALAMI dataset. 89

4.1 A tensor, seen with its 3 modes and in slices. 96
4.2 Order for unfolding and vectorization. 96
4.3 Nonnegative Tucker Decomposition of tensor $\mathcal{X}$ in factor matrices W, H, Q, and core tensor $\mathcal{G}$, with their dimensions. 99
4.4 TFB tensor 111
4.5 A toy example of transcription using NMF, adapted from [WMC22]. 113
4.6 A simple drum pattern, taken as an example for NMTF 114
4.7 Drum pattern from Figure 4.6, decomposed in both NMF and NMTF models. Matrix H is composed of 5 different barwise rhythmic templates, but only 2 are actually used ($H_{2:}$ and $H_{3:}$), presenting H as a dictionary of rhythmic patterns. This is useful in the interpretation of NTD, as matrices W and H can be interpreted as dictionaries of pitch-related and duration-related information (at barscale) in the song. 115
4.8 NTD example on the chromagram of *Come Together* by The Beatles, with dimensions $F' = 12, S' = 12, B' = 10$, and $W = I_{12}$. Columns of the H and Q matrices (and, accordingly, slices of $\mathcal{G}$) are reordered for visualization purposes. 117
4.9 Different autosimilarities on the song *POP01* of RWC Pop, computed on the feature (Barwise TF matrix, top) and on the Q matrix resulting of the NTD (bottom). Our particular example for NTD was taken for the chromagram feature, with Euclidean-NTD and by setting $W = I_{12}, S' = 16$ and $B' = 16$. 120

5.1 Barwise TF matrix. 139

5.2 Examples of Cosine autosimilarities of both the NMF and PCA, compared with the raw Barwise TF Cosine autosimilarity. These autosimilarities are computed on the NNLMS of the song *Pop01* of RWC Pop, with $B' = 24$. . 143
5.3 Segmentation results for PCA, on the different autosimilarities. 145
5.4 Segmentation results for Euclidean-NMF, on the different autosimilarities. 145
5.5 Segmentation results for KL-NMF, on the different autosimilarities. 146
5.6 Segmentation results with the Cosine similarity, computed on the Q matrix obtained from PCA and Euclidean-NMF, compared with the segmentation results on the raw Barwise TF and with Euclidean-NTD. 147
5.7 Results for NMF with different loss functions, on the Cosine similarity. . . 147
5.8 Schematic representation of the general SSAE. 152
5.9 Architecture of the Conv SSAE. 154
5.10 Example of Cosine autosimilarity of the SSAE, compared with the Barwise TF-, NMF- and PCA-based Cosine autosimilarities. These autosimilarities are computed on the NNLMS of the song *Pop01* of RWC Pop, with $B' = 24$.157
5.11 Segmentation results according to the feature and the similarity function, for the Euclidean-SSAE. The scores above the boxplots represent the median of averaged F measures according to the five different runs. 159
5.12 Median segmentation results according to the feature, for the Cosine autosimilarity of the Euclidean-SSAE and the previous barwise compression techniques. 159
5.13 Segmentation results according to the nonnegative features, for the Cosine autosimilarities of the Euclidean-, KL- and IS-SSAEs. 160
5.14 Segmentation results according to the different similarity functions features, for the Euclidean-SSAE, on the NNLM and MFCC spectrograms. 161
5.15 Plots of both Sigmoid and hyperbolic tangent functions. 163
5.16 Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, without activation function on the latent space, for the Log Mel spectrogram of the song *POP01*. 164
5.17 Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, with the Softmax activation function on the latent space, for the Log Mel spectrogram of the song *POP01*. 164

5.18 Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, with the Sigmoid activation function on the latent space, for the Log Mel spectrogram of the song *POP01*. 165

5.19 Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, with the hyperbolic tangent activation function on the latent space, for the Log Mel spectrogram of the song *POP01*. 165

5.20 Schematic representation of the Triplet SSAE. 167

5.21 Q matrix and Cosine autosimilarity of the latent representation of the Euclidean-SSAE, without activation function on the latent space, for the Log Mel spectrogram of the song *POP01* (shown again for comparison with the Triplet Loss model). 168

5.22 Q matrix and Cosine autosimilarity of the latent representation of the Triplet Conv SSAE, optimized subject to the Euclidean distance, for the Log Mel spectrogram of the song *POP01*. 169

6.1 Schematic representation of an AE-NMF. 177

6.2 Schematic representation of an AE-NTD. The encoder can be of any shape (fully-connected, convolutional, ...) 178

6.3 Architecture of the AE-NTD. 180

6.4 General matrices W_{RWC} and H_{RWC} computed from the Euclidean-NTD of the NNLM spectrograms of the whole RWC Pop dataset. 184

6.5 Cosine autosimilarities for the Euclidean-NTD and Euclidean-AE-NTD, computed on the NNLMS of the song *POP01*, for $F' = S' = B' = 24$. . . . 186

6.6 Segmentation results for the Euclidean AE-NTD, depending on the initialization of the decoder, compared with the NTD. 187

6.7 Cosine autosimilarities for the KL-NTD and KL-AE-NTD, computed on the NNLMS of the song *POP01*, for $F' = S' = B' = 24$. 189

6.8 Segmentation results for the KL AE-NTD, depending on the initialization of the decoder, compared with the NTD. 190

B.1 Architecture of the FC SSAE. 233

B.2 Segmentation results for different batch sizes, for the Conv SSAE, on the Cosine autosimilarity of the Log Mel representation. 235

B.3 Segmentation results for different batch sizes, for the FC SSAE, with different features and the Cosine and Covariance autosimilarities (the RBF autosimilarity is not presented, because results were mainly similar to those of Covariance). The different conditions seem to lead to different conclusions, complicating the decision-making process. 236
B.4 Architecture of the FC SSAE with batch normalization layers. 237
B.5 Segmentation results for the Conv network, with and without batch normalization layers. Results are presented on the Cosine and Covariance autosimilarities, with different features (namely Log Mel and MFCC). 238
B.6 Segmentation results for the FC network, with and without batch normalization layers. Results are presented on the Log Mel feature, with different autosimilarities. 239
B.7 Cosine autosimilarities of the different SSAE, with and without batch normalization layers. 240

C.1 Segmentation results for the KL AE-NTD, depending on the initialization of the decoder, compared with the NTD. 241
C.2 Q matrices (latent projections for the AE-NTD) of the General Init AE-NTD compared with those of the Random Init AE-NTD and the NTD, computed on the NNLMS of the song *POP01*, for $F' = S' = B' = 24$. . . . 243

List of Tables

2.1 Metrics when aligning the references on the downbeats (compared to the original annotations). The SALAMI dataset is restricted to the test subset, as detailed in Appendix A.2, and the two sets of annotations are presented. 63

3.1 Results when computing different autosimilarities on the Barwise TF matrix, on the RWC Pop dataset. (Full kernel, no penalty function.) 82

3.2 Results when computing different autosimilarities on the Barwise TF matrix, on the SALAMI dataset. (Full kernel, no penalty function.) 82

3.3 Segmentation results depending on the penalty function, for the RWC Pop dataset, with the RBF autosimilarity and the 7-bands kernel. 87

3.4 Segmentation results depending on the penalty function, for the SALAMI dataset, with the RBF autosimilarity and the 15-bands kernel. 87

4.1 Comparing results between Barwise TF-based and NTD-based Cosine autosimilarities, on the chromagram. 122

4.2 Results when computing different autosimilarities on Q, on the chromagram. 122

4.3 Segmentation results on the Cosine autosimilarity of Q with Euclidean-, KL- and IS-NTD, on the chromagram. 123

4.4 Segmentation results when computing Euclidean-, KL- and IS-NTD on Cosine similarity. 124

4.5 Segmentation results for KL-NTD with the different autosimilarities. . . . 125

4.6 Comparison of the best-performing NTD (Cosine similarity of Q obtained with KL-NTD on the NNLM spectrogram) with the best-performing Barwise TF on the same feature (RBF on the NNLM spectrogram), the global best-performing Barwise TF (RBF on the Log Mel spectrogram), and the State-of-the-Art methods. 125

4.7 Results of Euclidean-NTD with dimensions fitted by cross-validation (previous scenario) and in the oracle condition, on the chromagram. 126

4.8 Average and standard-deviation of the SDR scores for the 10 patterns obtained from NTD on the STFT of the song *Come Together*. Note that two Euclidean-NTD patterns resulted in a mask with every entry equal to one, and these patterns were discarded in the computation of SDR (the spectrogram of the pattern is exactly the spectrogram of the bar in this scenario, which is not informative). 133
4.9 SDR when reconstructing the signal of the entire song from the whole NTD, for *Come Together*. 134

5.1 Comparison of the best-performing linear compression methods. 147
5.2 Segmentation results of NMF and PCA on SALAMI, with the Cosine autosimilarity. *We recall that the Barwise TF condition means that there is no compression. 148
5.3 Comparison of the SSAEs with the best-performing methods, on RWC Pop. 160
5.4 Segmentation results of SSAEs on SALAMI with the Cosine autosimilarity, compared to the best other segmentation methods. *The Barwise TF condition means that there is no compression. 161
5.5 Comparison of all best-performing methods. 170

6.1 Average elementwise reconstruction error, in term of Euclidean distance, for the different techniques computed on the RWC Pop dataset. 188
6.2 Average elementwise reconstruction error, in term of KL-divergence, for the different intializations. 188
6.3 SDR results, both at the song scale and averaged for the different patterns, for the Euclidean-NTD and Euclidean-AE-NTD. 192
6.4 SDR results, both at the song scale and averaged for the different patterns, for the KL-NTD and KL-AE-NTD. 192
6.5 Comparison of the segmentation scores of the Conv SSAEs and the Random Init AE-NTDs, for the NNLMS and the RWC Pop dataset. 193

C.1 Reconstruction error, average for a bar in the song, in term of KL-divergence, for the different techniques. 242

BIBLIOGRAPHY

[Ale+22] Boian Alexandrov, Derek F DeSantis, Gianmarco Manzini, and Erik W Skau, « Nonnegative canonical tensor decomposition with linear constraints: nnCANDELINC », *in*: *Numerical Linear Algebra with Applications* (2022), e2443.

[AM01] Jean-Julien Aucouturier and Sandler Mark, « Segmentation of musical signals using hidden Markov models », *in*: *Proc. 110th Convention of the Audio Engineering Society*, Citeseer, 2001.

[And58] Theodore W Anderson, *An introduction to multivariate statistical analysis*, tech. rep., 1958.

[Bal10] Radu Balan, « On signal reconstruction from its spectrogram », *in*: *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, IEEE, 2010, pp. 1–4.

[Bas+98] Ayanendranath Basu, Ian R Harris, Nils L Hjort, and MC Jones, « Robust and Efficient Estimation by Minimising a Density Power Divergence », *in*: *Biometrika* 85.*3* (1998), pp. 549–559.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent, « Representation learning: A review and new perspectives », *in*: *IEEE transactions on pattern analysis and machine intelligence* 35.*8* (2013), pp. 1798–1828.

[Bec17] Amir Beck, *First-order methods in optimization*, SIAM, 2017.

[Ben+03] Laurent Benaroya, Lorcan M Donagh, Frédéric Bimbot, and Rémi Gribonval, « Non negative sparse representation for Wiener based source separation with a single sensor », *in*: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, 2003.

[Ber+11] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere, « The million song dataset », *in*: (2011).

[Ber+20] Jacopo de Berardinis, Michalis Vamvakaris, Angelo Cangelosi, and Eduardo Coutinho, « Unveiling the hierarchical structure of music by multi-resolution community detection », *in*: *Transactions of the International Society for Music Information Retrieval* 3.*1* (2020), pp. 82–97.

[Ber09] Nancy Bertin, « Les factorisations en matrices non-négatives. Approches contraintes et probabilistes, application à la transcription automatique de musique polyphonique. », PhD thesis, Télécom ParisTech, 2009.

[BG08] Christos Boutsidis and Efstratios Gallopoulos, « SVD based initialization: A head start for nonnegative matrix factorization », *in*: *Pattern recognition* 41.*4* (2008), pp. 1350–1362.

[BH89] Pierre Baldi and Kurt Hornik, « Neural networks and principal component analysis: Learning from examples without local minima », *in*: *Neural networks* 2.*1* (1989), pp. 53–58.

[BHP20] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet, *Deep learning techniques for music generation*, vol. 1, Springer, 2020.

[Bim+10] Frédéric Bimbot, Olivier Le Blouch, Gabriel Sargent, and Emmanuel Vincent, « Decomposition into autonomous and comparable blocks: a structural description of music pieces », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

[Bim+12] Frédéric Bimbot, Emmanuel Deruty, Gabriel Sargent, and Emmanuel Vincent, « Semiotic structure labeling of music pieces: Concepts, methods and annotation conventions », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[BKW16] Sebastian Böck, Florian Krebs, and Gerhard Widmer, « Joint Beat and Downbeat Tracking with Recurrent Neural Networks », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, New York City, 2016, pp. 255–261.

[BMK06] Michael J Bruderer, Martin F McKinney, and Armin Kohlrausch, « Structural boundary perception in popular music », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2006, pp. 198–201.

[Böc+16] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer, « Madmom: A new python audio and music signal processing library », *in*: *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1174–1178.

[BP92] Judith C Brown and Miller S Puckette, « An efficient algorithm for the calculation of a constant Q transform », *in*: *The Journal of the Acoustical Society of America* 92.*5* (1992), pp. 2698–2701.

[BST14] Jérôme Bolte, Shoham Sabach, and Marc Teboulle, « Proximal alternating linearized minimization for nonconvex and nonsmooth problems », *in*: *Mathematical Programming* 146.*1* (2014), pp. 459–494.

[BW01] Mark A Bartsch and Gregory H Wakefield, « To catch a chorus: Using chroma-based representations for audio thumbnailing », *in*: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*, IEEE, 2001, pp. 15–18.

[CC05] Ching-Hua Chuan and Elaine Chew, « Polyphonic audio key finding using the spiral array CEG algorithm », *in*: *2005 IEEE International Conference on Multimedia and Expo*, IEEE, 2005, pp. 21–24.

[CCS19] Jonah Casebeer, Michael Colomb, and Paris Smaragdis, « Deep tensor factorization for spatially-aware scene decomposition », *in*: *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2019, pp. 180–184.

[Che+16] Tian Cheng, Matthias Mauch, Emmanouil Benetos, Simon Dixon, et al., « An attack/decay model for piano transcription », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[Che02] Elaine Chew, « The spiral array: An algorithm for determining key boundaries », *in*: *International Conference on Music and Artificial Intelligence*, Springer, 2002, pp. 18–31.

[Cic+09] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multiway data analysis and blind source separation*, John Wiley & Sons, 2009.

[CLS15] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi, « Phase retrieval via Wirtinger flow: Theory and algorithms », *in*: *IEEE Transactions on Information Theory* 61.*4* (2015), pp. 1985–2007.

[CM63] Grosvenor Cooper and Leonard B Meyer, *The rhythmic structure of music*, University of Chicago press, 1963.

[Coh15] Jeremy E Cohen, « About notations in multiway array processing », *in*: *arXiv preprint arXiv:1511.01306* (2015).

[Cor+09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein, *Introduction to algorithms*, 3rd ed., MIT press, 2009.

[CSG18] Tian Cheng, Jordan BL Smith, and Masataka Goto, « Music structure boundary detection and labelling by a deconvolution of path-enhanced self-similarity matrix », *in*: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 106–110.

[CZA07] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari, « Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization », *in*: *International Conference on Independent Component Analysis and Signal Separation*, Springer, 2007, pp. 169–176.

[DDV00] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle, « A multilinear singular value decomposition », *in*: *SIAM journal on Matrix Analysis and Applications* 21.*4* (2000), pp. 1253–1278.

[DGS21] Pierre De Handschutter, Nicolas Gillis, and Xavier Siebert, « A survey on deep matrix factorizations », *in*: *Computer Science Review* 42 (2021), p. 100423.

[DH03] Roger B Dannenberg and Ning Hu, « Pattern discovery techniques for music audio », *in*: *Journal of New Music Research* 32.*2* (2003), pp. 153–163.

[Din+06] Chris Ding, Tao Li, Wei Peng, and Haesun Park, « Orthogonal nonnegative matrix t-factorizations for clustering », *in*: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 126–135.

[Dow08] J Stephen Downie, « The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research », *in*: *Acoustical Science and Technology* 29.*4* (2008), pp. 247–255.

[DS14] Sander Dieleman and Benjamin Schrauwen, « End-to-end learning for music audio », *in*: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 6964–6968.

[DV16] Vincent Dumoulin and Francesco Visin, « A guide to convolution arithmetic for deep learning », *in*: *arXiv preprint arXiv:1603.07285* (2016).

[EHM16] Yonina C Eldar, Nethaniel Hammen, and Dustin G Mixon, « Recent advances in phase retrieval [lecture notes] », *in*: *IEEE signal processing magazine* 33.*5* (2016), pp. 158–162.

[Emi+11] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann, « Subjective and objective quality assessment of audio source separation », *in*: *IEEE Transactions on Audio, Speech, and Language Processing* 19.*7* (2011), pp. 2046–2057.

[Eng+17] Jesse H Engel et al., « Neural audio synthesis of musical notes with wavenet autoencoders », *in*: *Int. Conf. Machine Learning*, PMLR, 2017, pp. 1068–1077.

[Ewe+16] Sebastian Ewert, Siying Wang, Meinard Müller, and Mark Sandler, « Score-informed identification of missing and extra notes in piano recordings », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[FBD09] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu, « Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis », *in*: *Neural computation* 21.*3* (2009), pp. 793–830.

[FI11] Cédric Févotte and Jérôme Idier, « Algorithms for nonnegative matrix factorization with the β-divergence », *in*: *Neural computation* 23.*9* (2011), pp. 2421–2456.

[FJ12] Derry Fitzgerald and Rajesh Jaiswal, « On the use of masking filters in sound source separation », *in*: *Proc. of 15th International Conference on Digital Audio Effects,(DAFX12)*, 2012.

[Foo00] Jonathan Foote, « Automatic audio segmentation using a measure of audio novelty », *in*: *2000 IEEE Int. Conf. Multimedia and Expo. ICME2000. Proc. Latest Advances in the Fast Changing World of Multimedia*, IEEE, 2000, pp. 452–455.

[Fue+19] Magdalena Fuentes, Brian McFee, Hélène C Crayencour, Slim Essid, and Juan Pablo Bello, « A music structure informed downbeat tracking system using skip-chain conditional random fields and deep learning », *in*: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 481–485.

[Gao+17] Lufei Gao, Li Su, Yi-Hsuan Yang, and Tan Lee, « Polyphonic piano note transcription with non-negative matrix factorization of differential spectrogram », *in*: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 291–295.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, `http://www.deeplearningbook.org`, MIT Press, 2016.

[Gem+17] Jort F Gemmeke et al., « Audio Set: An ontology and human-labeled dataset for audio events », *in*: *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[GG12] Nicolas Gillis and François Glineur, « Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization », *in*: *Neural computation* 24.*4* (2012), pp. 1085–1105.

[GGL16] Mathieu Giraud, Richard Groult, and Florence Levé, « Computational analysis of musical form », *in*: *Computational Music Analysis*, Springer, 2016, pp. 113–136.

[Gil20] Nicolas Gillis, *Nonnegative Matrix Factorization*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2020.

[GL84] Daniel Griffin and Jae Lim, « Signal estimation from modified short-time Fourier transform », *in*: *IEEE Transactions on acoustics, speech, and signal processing* 32.*2* (1984), pp. 236–243.

[GLT21] Nicolas Gillis, Valentin Leplat, and Vincent Tan, « Distributionally robust and multi-objective nonnegative matrix factorization », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[GM90] Brian R Glasberg and Brian CJ Moore, « Derivation of auditory filter shapes from notched-noise data », *in*: *Hearing research* 47.*1-2* (1990), pp. 103–138.

[Got+02] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka, « RWC Music Database: Popular, Classical and Jazz Music Databases », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2002, pp. 287–288.

[Got+06] Masataka Goto et al., « AIST Annotation for the RWC Music Database », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2006, pp. 359–360.

[Got03] Masataka Goto, « A chorus-section detecting method for musical audio signals », *in*: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, 2003.

[GS15a] Mathieu Giraud and Sławek Staworko, « Modeling musical structure with parametric grammars », *in*: *International conference on mathematics and computation in music*, Springer, 2015, pp. 85–96.

[GS15b] Thomas Grill and Jan Schlüter, « Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 531–537.

[Gui17] Corentin Guichaoua, « Modèles de compression et critères de complexité pour la description et l'inférence de structure musicale », PhD thesis, Université Rennes 1, 2017.

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, « Delving deep into rectifiers: Surpassing human-level performance on imagenet classification », *in*: *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[Her+17] Shawn Hershey et al., « CNN architectures for large-scale audio classification », *in*: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 131–135.

[HLW14] John R Hershey, Jonathan Le Roux, and Felix Weninger, « Deep unfolding: Model-based inspiration of novel deep architectures », *in*: *arXiv preprint arXiv:1409.2574* (2014).

[Hot33] Harold Hotelling, « Analysis of a complex of statistical variables into principal components », *in*: *Journal of educational psychology* 24.*6* (1933), p. 417.

[HWL21] Yun-Ning Hung, Gordon Wichern, and Jonathan Le Roux, « Transcription is all you need: Learning to separate musical mixtures with score as supervision », *in*: *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 46–50.

[IS15] Sergey Ioffe and Christian Szegedy, « Batch normalization: Accelerating deep network training by reducing internal covariate shift », *in*: *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[Jen06] Kristoffer Jensen, « Multiple scale music segmentation using rhythm, timbre, and harmony », *in*: *EURASIP Journal on Advances in Signal Processing* 2007 (2006), pp. 1–11.

[Jol02] Ian T Jolliffe, *Principal component analysis for special types of data*, Springer, 2002.

[KB09] Tamara G Kolda and Brett W Bader, « Tensor decompositions and applications », *in*: *SIAM review* 51.*3* (2009), pp. 455–500.

[KB14] Diederik P Kingma and Jimmy Ba, « Adam: A method for stochastic optimization », *in*: *arXiv preprint arXiv:1412.6980* (2014).

[KC07] Yong-Deok Kim and Seungjin Choi, « Nonnegative tucker decomposition », *in*: *2007 IEEE conference on computer vision and pattern recognition*, IEEE, 2007, pp. 1–8.

[KCC08] Yong-Deok Kim, Andrzej Cichocki, and Seungjin Choi, « Nonnegative Tucker decomposition with alpha-divergence », *in*: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2008, pp. 1829–1832.

[KKV13] Joonas Kauppinen, Anssi Klapuri, and Tuomas Virtanen, « Music self-similarity modeling using augmented nonnegative matrix factorization of block and stripe patterns », *in*: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, IEEE, 2013, pp. 1–4.

[Kos+19] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic, « TensorLy: Tensor Learning in Python », *in*: *Journal of Machine Learning Research* 20.*26* (2019).

[KS10] Florian Kaiser and Thomas Sikora, « Music Structure Discovery in Popular Music using Non-negative Matrix Factorization », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 429–434.

[KW16] Filip Korzeniowski and Gerhard Widmer, « Feature learning for chord recognition: The deep chroma extractor », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[Law88] Bo Lawergren, « The origin of musical instruments and sounds », *in*: *Anthropos* (1988), pp. 31–45.

[LB17] Corentin Louboutin and Frédéric Bimbot, « Modeling the multiscale structure of chord sequences using polytopic graphs », *in*: *18th International Society for Music Information Retrieval Conference*, 2017.

[LC00] Beth Logan and Stephen Chu, « Music summarization using key phrases », *in*: *2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2000.

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, « Gradient-based learning applied to document recognition », *in*: *Proceedings of the IEEE* 86.*11* (1998), pp. 2278–2324.

[Lef12] Augustin Lefèvre, « Dictionary learning methods for single-channel source separation », PhD thesis, École normale supérieure de Cachan-ENS Cachan, 2012.

[LH95] Charles L Lawson and Richard J Hanson, *Solving least squares problems*, SIAM, 1995.

[LHW15] Jonathan Le Roux, John R Hershey, and Felix Weninger, « Deep NMF for speech separation », *in*: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 66–70.

[LS06] Mark Levy and Mark Sandler, « New methods in structural segmentation of musical audio », *in*: *2006 14th European Signal Processing Conference*, IEEE, 2006.

[LS08] Mark Levy and Mark Sandler, « Structural segmentation of musical audio by constrained clustering », *in*: *IEEE transactions on audio, speech, and language processing* 16.*2* (2008), pp. 318–326.

[LS99] Daniel D Lee and H Sebastian Seung, « Learning the parts of objects by non-negative matrix factorization », *in*: *Nature* 401.*6755* (1999), pp. 788–791.

[Luk08] Hanna M Lukashevich, « Towards Quantitative Measures of Evaluating Song Segmentation. », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2008, pp. 375–380.

[LWZ04] Lie Lu, Muyuan Wang, and Hong-Jiang Zhang, « Repeating pattern discovery and structure analysis from acoustic music data », *in*: *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, 2004, pp. 275–282.

[Mac67] James B MacQueen, « Classification and analysis of multivariate observations », *in*: *5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[Mae19] Akira Maezawa, « Music boundary detection based on a hybrid deep model of novelty, homogeneity, repetition and duration », *in*: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 206–210.

[Mag16] Paul Magron, « Reconstruction de phase par modèles de signaux: application à la séparation de sources audio », PhD thesis, Télécom ParisTech, 2016.

[Mar+20] Axel Marmoret, Jérémy E Cohen, Nancy Bertin, and Frédéric Bimbot, « Uncovering Audio Patterns in Music with Nonnegative Tucker Decomposition for Structural Segmentation », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 788–794.

[Mar+22] Axel Marmoret, Florian Voorwinden, Valentin Leplat, Jérémy E Cohen, and Frédéric Bimbot, « Nonnegative Tucker Decomposition with Beta-divergence for Music Structure Analysis of audio signals », *in*: *GRETSI* (2022).

[Mar12] Ivan Markovsky, *Low rank approximation: algorithms, implementation, applications*, vol. 906, Springer, 2012.

[Mau+09] Matthias Mauch et al., « OMRAS2 metadata project 2009 », *in*: *Proc. of 10th International Conference on Music Information Retrieval*, 2009, p. 1.

[MC20] Axel Marmoret and Jérémy Cohen, *nn_fac: Nonnegative Factorization techniques toolbox*, 2020, URL: `https://gitlab.inria.fr/amarmore/nonnegative-factorization`.

[MCB21] Axel Marmoret, Jérémy Cohen, and Frédéric Bimbot, *MusicOnPolytopes*, 2021, URL: `https://gitlab.inria.fr/amarmore/musiconpolytopes`.

[MCB22a] Axel Marmoret, Jérémy Cohen, and Frédéric Bimbot, *as_seg: module for computing and segmenting autosimilarity matrices*, 2022, URL: `https://gitlab.inria.fr/amarmore/autosimilarity_segmentation`.

[MCB22b] Axel Marmoret, Jérémy Cohen, and Frédéric Bimbot, *BarMusComp: module for computing barwise compressed representations of music*, 2022, URL: `https://gitlab.inria.fr/amarmore/barmuscomp`.

[MCB22c] Axel Marmoret, Jérémy E Cohen, and Frédéric Bimbot, « Barwise Compression Schemes for Audio-Based Music Structure Analysis », *in*: *19th Sound and Music Computing Conference, SMC 2022*, Sound and music Computing network, 2022.

[MCB22d] Axel Marmoret, Jérémy E Cohen, and Frédéric Bimbot, « Convolutive Block-Matching Segmentation Algorithm with Application to Music Structure Analysis », *in*: *arXiv preprint arXiv:2210.15356* (2022).

[McC19] Matthew C McCallum, « Unsupervised learning of deep features for music segmentation », *in*: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 346–350.

[McF+17] Brian McFee, Oriol Nieto, Morwaread M Farbood, and Juan Pablo Bello, « Evaluating hierarchical structure in music annotations », *in*: *Frontiers in psychology* 8 (2017), p. 1337.

[McF+21] Brian McFee et al., « librosa/librosa: 0.8.1rc2 », *in*: Zenodo, May 2021, DOI: `10.5281/zenodo.4792298`, URL: `https://doi.org/10.5281/zenodo.4792298`.

[ME14a] Brian McFee and Dan Ellis, « Analyzing Song Structure with Spectral Clustering », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 405–410.

[ME14b] Brian McFee and Daniel PW Ellis, « Learning to segment songs with ordinal linear discriminant analysis », *in*: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 5197–5201.

[Mes+15] Annamaria Mesaros, Toni Heittola, Onur Dikmen, and Tuomas Virtanen, « Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations », *in*: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2015, pp. 151–155.

[MGF21] Arthur Marmin, José Henrique de Morais Goulart, and Cédric Févotte, « Joint Majorization-Minimization for Nonnegative Matrix Factorization with the β-divergence », *in*: *arXiv preprint arXiv:2106.15214* (2021).

[MHA08] Morten Mørup, Lars Kai Hansen, and Sidse M Arnfred, « Algorithms for sparse nonnegative Tucker decompositions », *in*: *Neural computation* 20.*8* (2008), pp. 2112–2131.

[MKC05] Meinard Müller, Frank Kurth, and Michael Clausen, « Audio Matching via Chroma-Based Statistical Features », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2005.

[MLE21] Vishal Monga, Yuelong Li, and Yonina C Eldar, « Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing », *in*: *IEEE Signal Processing Magazine* 38.*2* (2021), pp. 18–44.

[MND09] Matthias Mauch, Katy C Noland, and Simon Dixon, « Using Musical Structure to Enhance Automatic Chord Transcription », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2009, pp. 231–236.

[Mül15] Meinard Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*, vol. 5, Springer, 2015.

[NB14] Oriol Nieto and Juan Pablo Bello, « Music segment similarity using 2d-fourier magnitude coefficients », *in*: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 664–668.

[NB16] Oriol Nieto and Juan Pablo Bello, « Systematic exploration of computational music structure research », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 547–553.

[Neb+21] Benjamin T Nebgen, Raviteja Vangara, Miguel A Hombrados-Herrera, Svetlana Kuksova, and Boian S Alexandrov, « A neural network for determination of latent dimensionality in non-negative matrix factorization », *in*: *Machine Learning: Science and Technology* 2.*2* (2021), p. 025012.

[Nee22] Adam Neely, *This BPM is trash, and here's why*, `https://www.youtube.com/watch?v=nUHEPmg0sPo`, Feb. 2022.

[Nie+19] Oriol Nieto, Matthew McCallum, Matthew EP Davies, Andrew Robertson, Adam M Stark, and Eran Egozy, « The Harmonix Set: Beats, Downbeats, and Functional Segment Annotations of Western Popular Music », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 565–572.

[Nie+20] Oriol Nieto et al., « Audio-Based Music Structure Analysis: Current Trends, Open Challenges, and Applications », *in*: *Trans. Int. Soc. for Music Information Retrieval* 3.*1* (2020).

[NJ13] Oriol Nieto and Tristan Jehan, « Convex non-negative matrix factorization for automatic music structure identification », *in*: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2013, pp. 236–240.

[NLV16] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent, « Multichannel audio source separation with deep neural networks », *in*: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.*9* (2016), pp. 1652–1664.

[NVG20] Nicolas Nadisic, Arnaud Vandaele, and Nicolas Gillis, « A homotopy-based algorithm for sparse multiple right-hand sides nonnegative least squares », *in*: *arXiv preprint arXiv:2011.11066* (2020).

[OF09] Alexey Ozerov and Cédric Févotte, « Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation », *in*: *IEEE Transactions on Audio, Speech, and Language Processing* 18.*3* (2009), pp. 550–563.

[OH05] Bee Suan Ong and Perfecto Herrera, « Semantic segmentation of music audio », *in*: *Proceedings of the 2005 International Computer Music Conference*, Computer Music Association, 2005, p. 61.

[OK85] Erkki Oja and Juha Karhunen, « On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix », *in*: *Journal of mathematical analysis and applications* 106.*1* (1985), pp. 69–84.

[Oku+17] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima, « Embedding-based news recommendation for millions of users », *in*: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1933–1942.

[Ong+06] Bee Suan Ong et al., *Structural analysis and segmentation of music signals*, Citeseer, 2006.

[OP14] Ken O'Hanlon and Mark D Plumbley, « Polyphonic piano transcription using non-negative matrix factorisation with group sparsity », *in*: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 3112–3116.

[OSh87] Douglas O'Shaughnessy, *Speech communications: Human and machine (IEEE)*, Universities press, 1987.

[Pas+19] Adam Paszke et al., « Pytorch: An imperative style, high-performance deep learning library », *in*: *Advances in neural information processing systems* 32 (2019).

[PC11] Anh Huy Phan and Andrzej Cichocki, « Extended HALS algorithm for non-negative Tucker decomposition and its applications for multiway analysis and classification », *in*: *Neurocomputing* 74.*11* (2011), pp. 1956–1969.

[PD09] Geoffroy Peeters and Emmanuel Deruty, « Is music structure annotation multi-dimensional? A proposal for robust local music annotation », *in*: *Proc. of 3rd Workshop on Learning the Semantics of Audio Signals*, Citeseer, 2009, pp. 75–90.

[Pea01] Karl Pearson, « On lines of closes fit to system of points in space, London, E dinb », *in*: *Dublin Philos. Mag. J. Sci* 2 (1901), pp. 559–572.

[Ped+11] Fabian Pedregosa et al., « Scikit-learn: Machine Learning in Python », *in*: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Pee03] Geoffroy Peeters, « Deriving musical structures from signal analysis for music audio summary generation:"sequence" and "state" approach », *in*: *International Symposium on Computer Music Modeling and Retrieval*, Springer, 2003, pp. 143–166.

[Pee07] Geoffroy Peeters, « Sequence Representation of Music Structure Using Higher-Order Similarity Matrix and Maximum-Likelihood Approach », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 35–40.

[PKS22] Miguel Perez, Holger Kirchhoff, and Xavier Serra, « A Comparison of Pitch Chroma Extraction Algorithms », *in*: *19th Sound and Music Computing Conference, SMC 2022*, Sound and music Computing network, 2022.

[PLR02] Geoffroy Peeters, Amaury La Burthe, and Xavier Rodet, « Toward automatic music audio summary generation from signal analysis », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2002.

[PMK10] Jouni Paulus, Meinard Müller, and Anssi Klapuri, « State of the Art Report: Audio-Based Music Structure Analysis », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, 2010, pp. 625–636.

[PMW10] Marcus T Pearce, Daniel Müllensiefen, and Geraint A Wiggins, « Melodic grouping in music information retrieval: New methods and applications », *in*: *Advances in music information retrieval*, Springer, 2010, pp. 364–388.

[PR21] Geoffroy Peeters and Gaël Richard, « Deep Learning for Audio and Music », *in*: *Multi-faceted Deep Learning: Models and Data*, Springer, 2021, URL: `https://hal.telecom-paris.fr/hal-03153938`.

[Qia+20] Kaizhi Qian, Yang Zhang, Shiyu Chang, Mark Hasegawa-Johnson, and David Cox, « Unsupervised speech decomposition via triple information bottleneck », *in*: *International Conference on Machine Learning*, PMLR, 2020, pp. 7836–7846.

[Rab89] Lawrence R Rabiner, « A tutorial on hidden Markov models and selected applications in speech recognition », *in*: *Proceedings of the IEEE* 77.*2* (1989), pp. 257–286.

[Raf+14] Colin Raffel et al., « mir_eval: A transparent implementation of common MIR metrics », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[RHL13] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo, « A unified convergence analysis of block successive minimization methods for nonsmooth optimization », *in*: *SIAM Journal on Optimization* 23.*2* (2013), pp. 1126–1153.

[Roc+18] Fanny Roche, Thomas Hueber, Samuel Limier, and Laurent Girin, « Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models », *in*: *arXiv preprint arXiv:1806.04096* (2018).

[SB03] Paris Smaragdis and Judith C Brown, « Non-negative matrix factorization for polyphonic music transcription », *in*: *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, IEEE, 2003, pp. 177–180.

[SBV16] Gabriel Sargent, Frédéric Bimbot, and Emmanuel Vincent, « Estimating the structural segmentation of popular music pieces under regularity constraints », *in*: *IEEE/ACM Trans. Audio, Speech, and Language Processing* 25.*2* (2016), pp. 344–358.

[SC13] Jordan BL Smith and Elaine Chew, « Using quadratic programming to estimate feature relevance in structural analyses of music », *in*: *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 113–122.

[Ser+14] Joan Serra, Meinard Müller, Peter Grosche, and Josep Ll Arcos, « Unsupervised music structure annotation by time series structure features and segment similarity », *in*: *IEEE Transactions on Multimedia* 16.*5* (2014), pp. 1229–1240.

[SG16] Jordan BL Smith and Masataka Goto, « Using Priors to Improve Estimates of Music Structure », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 554–560.

[SG17] Jordan BL Smith and Masataka Goto, « Multi-Part Pattern Analysis: Combining Structure Analysis and Source Separation to Discover Intra-Part Repeated Sequences », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 716–723.

[SG18] Jordan BL Smith and Masataka Goto, « Nonnegative tensor factorization for source separation of loops in audio », *in*: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 171–175.

[Sha49] Claude E Shannon, « Communication in the presence of noise », *in*: *Proceedings of the IRE* 37.*1* (1949), pp. 10–21.

[She+15] Yoav Shechtman, Yonina C Eldar, Oren Cohen, Henry Nicholas Chapman, Jianwei Miao, and Mordechai Segev, « Phase retrieval with application to optical imaging: a contemporary overview », *in*: *IEEE signal processing magazine* 32.*3* (2015), pp. 87–109.

[SJK06] Yu Shiu, Hong Jeong, and C-C Jay Kuo, « Similarity matrix processing for music structure analysis », *in*: *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 2006, pp. 69–76.

[SKG19] Jordan BL Smith, Yuta Kawasaki, and Masataka Goto, « Unmixer: An Interface for Extracting and Remixing Loops », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 824–831.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin, « Facenet: A unified embedding for face recognition and clustering », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[SM06] Mikkel N Schmidt and Morten Mørup, « Nonnegative matrix factor 2-D deconvolution for blind single channel source separation », *in*: *International Conference on Independent Component Analysis and Signal Separation*, Springer, 2006, pp. 700–707.

[Smi+11] Jordan BL Smith, John A Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie, « Design and creation of a large-scale database of structural annotations », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, Miami, FL, 2011, pp. 555–560.

[SNB21] Justin Salamon, Oriol Nieto, and Nicholas J Bryan, « Deep Embeddings and Section Fusion Improve Music Segmentation », *in*: *IEEE Signal Processing Letters* 24.*3* (2021), pp. 279–283.

[ST04] Yoiti Suzuki and Hisashi Takeshima, « Equal-loudness-level contours for pure tones », *in*: *The Journal of the Acoustical Society of America* 116.*2* (2004), pp. 918–933.

[Ste93] Gilbert W Stewart, « On the early history of the singular value decomposition », *in*: *SIAM review* 35.*4* (1993), pp. 551–566.

[SV17] Paris Smaragdis and Shrikant Venkataramani, « A neural network alternative to non-negative audio models », *in*: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 86–90.

[SVN37] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman, « A scale for the measurement of the psychological magnitude pitch », *in*: *The journal of the acoustical society of america* 8.*3* (1937), pp. 185–190.

[TEF13] Ilias Theodorakopoulos, George Economou, and Spiros Fotopoulos, « Unsupervised music segmentation via multi-scale processing of compressive features' representation », *in*: *2013 18th International Conference on Digital Signal Processing (DSP)*, IEEE, 2013, pp. 1–6.

[TM19] Christopher J Tralie and Brian McFee, « Enhanced hierarchical music structure annotations via feature level similarity fusion », *in*: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 201–205.

[Tur+07] Douglas Turnbull, Gert RG Lanckriet, Elias Pampalk, and Masataka Goto, « A Supervised Approach for Detecting Boundaries in Music Using Difference Features and Boosting », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 51–54.

[UCN99] Srinivasan Umesh, Leon Cohen, and D Nelson, « Fitting the mel scale », *in*: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, 1999, pp. 217–220.

[Vav10] Stephen A Vavasis, « On the complexity of nonnegative matrix factorization », *in*: *SIAM Journal on Optimization* 20.*3* (2010), pp. 1364–1377.

[VBB09] Emmanuel Vincent, Nancy Bertin, and Roland Badeau, « Adaptive harmonic spectral decomposition for multiple pitch estimation », *in*: *IEEE Transactions on Audio, Speech, and Language Processing* 18.*3* (2009), pp. 528–537.

[VGF06] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte, « Performance measurement in blind audio source separation », *in*: *IEEE transactions on audio, speech, and language processing* 14.*4* (2006), pp. 1462–1469.

[Via+21] Pierre-Hugo Vial, Paul Magron, Thomas Oberlin, and Cédric Févotte, « Phase retrieval with Bregman divergences and application to audio signal recovery », *in*: *IEEE Journal of Selected Topics in Signal Processing* 15.*1* (2021), pp. 51–64.

[Vir07] Tuomas Virtanen, « Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria », *in*: *IEEE Transactions on Audio, Speech, and Language Processing* 15.*3* (2007), pp. 1066–1074.

[VOM21] Igor Vatolkin, Fabian Ostermann, and Meinard Müller, « An evolutionary multi-objective feature selection approach for detecting music segment boundaries of specific types », *in*: *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 1061–1069.

[VSS17] Shrikant Venkataramani, Cem Subakan, and Paris Smaragdis, « Neural network alternatives toconvolutive audio models for source separation », *in*: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2017, pp. 1–6.

[VTS20] Shrikant Venkataramani, Efthymios Tzinis, and Paris Smaragdis, « End-to-end non-negative autoencoders for sound source separation », *in*: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 116–120.

[Wan+21] Ju-Chiang Wang, Jordan BL Smith, Wei-Tsung Lu, and Xuchen Song, « Supervised Metric Learning for Music Structure Feature », *in*: *arXiv preprint arXiv:2110.09000* (2021).

[War70] W Dixon Ward, « Musical perception », *in*: *Foundations of modern auditory theory*, vol. 1, 1970, pp. 407–447.

[WB10] Ron J Weiss and Juan Pablo Bello, « Identifying repeated patterns in music using sparse convolutive non-negative matrix factorization », *in*: *11th Int Society for Music Information Retrieval Conf*, Citeseer, 2010, pp. 123–128.

[WHS22] Ju-Chiang Wang, Yun-Ning Hung, and Jordan BL Smith, « To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions », *in*: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 416–420.

[WKW16] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang, « A survey of transfer learning », *in*: *Journal of Big data* 3.*1* (2016), pp. 1–40.

[WM03] D Randall Wilson and Tony R Martinez, « The general inefficiency of batch training for gradient descent learning », *in*: *Neural networks* 16.*10* (2003), pp. 1429–1451.

[WMC22] Haoran Wu, Axel Marmoret, and Jérémy E Cohen, « Semi-Supervised Convolutive NMF for Automatic Music Transcription », *in*: *19th Sound and Music Computing Conference, SMC 2022*, Sound and music Computing network, 2022.

[WMD17] Cheng-i Wang, Gautham J Mysore, and Shlomo Dubnov, « Re-Visiting the Music Segmentation Problem with Crowdsourcing », *in*: *International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 738–744.

[Xu15] Yangyang Xu, « Alternating proximal gradient method for sparse nonnegative Tucker decomposition », *in*: *Mathematical Programming Computation* 7.*1* (2015), pp. 39–70.

[ZEH16] Zhenyao Zhu, Jesse H Engel, and Awni Hannun, « Learning multiscale features directly from waveforms », *in*: *arXiv preprint arXiv:1603.09509* (2016).

[ZF22] Rafał Zdunek and Krzysztof Fonał, « Incremental Nonnegative Tucker Decomposition with Block-Coordinate Descent and Recursive Approaches », *in*: *Symmetry* 14.*1* (2022), p. 113.

[Zho+15] Guoxu Zhou, Andrzej Cichocki, Qibin Zhao, and Shengli Xie, « Efficient nonnegative tucker decompositions: Algorithms and uniqueness », *in*: *IEEE Transactions on Image Processing* 24.*12* (2015), pp. 4990–5003.

Appendices

Appendix A

Experimental Details

This Appendix aims at precising experimental details, for clarity and reproduction purposes.

A.1 Features, in Details

This thesis studies and compares the different features presented in Section 2.2.4. Features are all computed using the *librosa* toolbox, and, unless specified below, spectrograms are computed using the default settings.

A.1.1 STFT

STFT are computed with windows containing $N = 2048$ samples. We compute the STFT as power spectrograms, *i.e.* using only the squared modulus $|.|^2$ of the STFT coefficients, and do not consider the phase information for our studies. Note though that an audio signal cannot be computed from a real-valued spectrogram solely, and requires phase information. This is further studied in Section 4.5.2.

A.1.2 Mel Spectrograms

Unless specified and for particular contexts (typically for audio reconstruction in Section 4.5.2), we prefer to use the Mel-rescaled STFT to the original STFT in this thesis. In details, the Mel scale are computed following [GS15b], *i.e.* using a Mel filter bank of 80 triangular filters, starting at 80Hz and ending at 16kHz. Still, we compute the coefficients of STFT as a power spectrogram instead of the magnitude spectrogram used in [GS15b].

Log Mel and NNLM spectrograms are computed on the aforementioned Mel spectrogram.

A.1.3 Chromagram

We insist on the fact that we use the CENS instead of the Chromagram, and abusively use the term of Chromagram. They are computed on 6 octaves, starting from G1 *i.e.* a lowest note of 98Hz. The smoothing window has a length of 82 frames. Final chromas vectors are normalized by the l^∞ norm.

A.1.4 MFCC

We compute the MFCC spectrograms with 32 bands, following [ME14b]. When computing the MFCC, we use the default *librosa* settings, thus, the Log Mel spectrogram used in the computation of the MFCC is not exactly the aforementioned Log Mel spectrogram.

A.2 Dataset in Learning/Test Paradigm

This thesis presents machine learning paradigms studying music. In that sense, while the techniques in themselves are unsupervised, some hyperparameters must be fixed to restrict the number of experiments and focus conclusions. Hereafter, we present the specific learning/test conditions used in this thesis, depending on the dataset (RWC Pop and SALAMI). Indeed, both datasets being composed of a different number of songs (respectively 100 and almost 1400), and following previous works in literature, we handle these datasets differently.

The RWC Pop dataset is handled in a two-fold cross-validation scheme, as in [Mar+20]: the RWC Pop dataset is divided in two subsets (songs with odd vs even ID number), which are alternatively used as learning and test datasets. When a subset is used for learning, all values in the considered range of hyperparameters are tested on this subset, and the hyperparameter leading to the best F-measure with both tolerances in average (*i.e.* average of $F_{0.5}$ and F_3) is used on the test subset. The final metrics are then computed as the average on both test subsets.

The SALAMI dataset is handled by dividing it in a learning and a test subset, following [GS15b]: approximately two-thirds of the data are treated as a way to learn hyperparameters (learning dataset), and the best hyperparameter is evaluated on the remaining of the data (test dataset, composed of 487 songs). The details of this repartition between learning and test dataset is available online [1], and tries to mimics the repartition used in

1. jan-schlueter.de/pubs/2014_ismir/

the MIREX contest (which is not publicly available). In particular, this test dataset contains 12 songs from RWC Pop, and the SALAMI dataset used in [GS15b] (v1.2) contains in total 15 songs from RWC Pop.

Appendix B

Technical Choices for the SSAE

B.1 FC SSAE

The "FC" network (FC SSAE) is a fully-connected neural network, with 3 hidden layers of respective sizes 128, B' and 128. This network is motivated by the work of [Roc+18]. Practically, the FC SSAE takes a vector as input, which is a barwise spectrogram, *i.e.* a row of the Barwise TF matrix. This barwise spectrogram is encoded by two fully-connected layers, resulting in a latent representation, finally decoded by the last hidden layer. It is represented in Figure B.1.

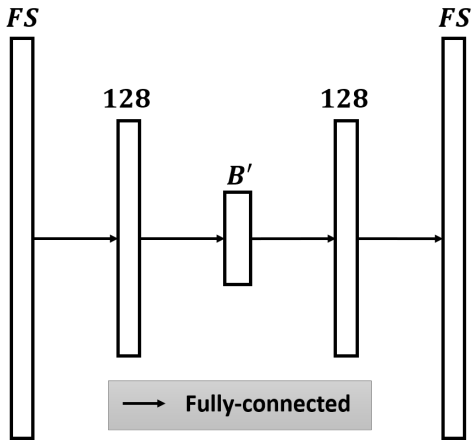


Figure B.1 – Architecture of the FC SSAE.

As an example, on an Intel® Core(TM) i7 CPU, decomposing the song *POP01* with $B' = 16$ for the Nonnegative Log Mel (NNLM) spectrogram takes approximately 1 minute for the FC SSAE, which is about 6 times faster than Conv SSAE.

B.2 Batch Size

The optimization paradigm is not only dependent of the choice of layers in the architecture (fully-connected and convolutional), but also of additional technical specifications. This section focuses on the size of the batch when processing the song, *i.e.* the number of samples to be presented to the network before backpropagating the error and updating the network' parameters. We compare the same networks when processing the entire song in a unique batch (batch of size B), and with sizes $\{8, 16, 24, 32, 64\}$.

As a first analysis, we use the task of structural segmentation as a proxy, to obtain quantitative conclusions. In addition, we notably study whether both FC and Conv SSAE obtain similar results.

In this first study, we only consider networks optimized subject to the Euclidean distance. As for experiments computed for NMF and PCA, we focus on the RWC Pop dataset for hyperparameter tuning, which contains fewer songs. In addition, the size of the latent space is fixed to 24, as for the Conv SSAE in Section 5.4.

For both networks, no particular batch size seems to stand out. For the Conv SSAE, results are similar between the different batch sizes, as presented for segmentation scores computed on the Cosine autosimilarity of the Log Mel spectrogram on Figure B.2 (we limit the Figure to this condition, but the trend is the same with the other features and similarities).

Conversely, for the FC SSAE, the batch size largely influences the results, but without a clear trend between conditions, as presented in Figure B.3. We explain this erratic behavior by the very high number of parameters compared to the number of samples (factor of 25 parameters for 1 sample, in order of magnitude).

With these results, we decide to fix the optimization strategy to a unique batch for each song, which represent the fastest condition in term of computation time: for the FC SSAE, on an Intel® Core(TM) i7 CPU, decomposing the song *POP01* with $B' = 16$ for the Nonnegative Log Mel (NNLM) spectrogram takes approximately 1 minute with a unique batch and almost 8 minutes with a batch size of 8. In addition, it reduces the risk of introducing bias when processing the song by batches.

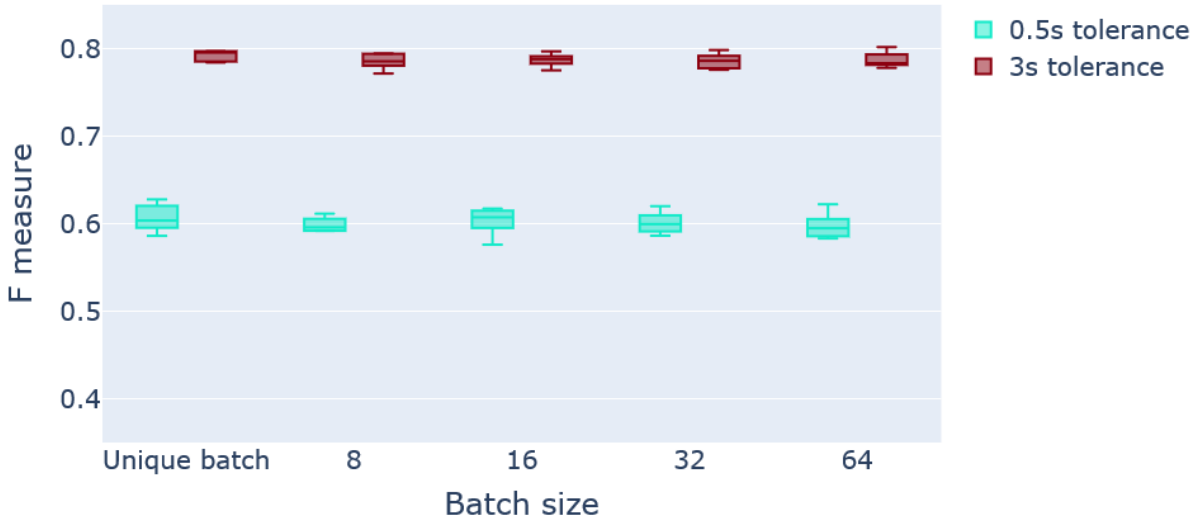


Figure B.2 – Segmentation results for different batch sizes, for the Conv SSAE, on the Cosine autosimilarity of the Log Mel representation.

B.3 Batch Normalization

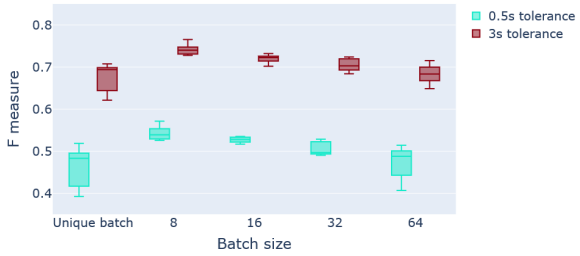
The erratic behavior of the FC SSAE must be handled, and this section introduces batch normalization layers [IS15], which are generally employed in neural networks in order to reduce the parameter space, and could be beneficial to that regard.

A batch normalization layer consists of two steps: firstly, normalizing the batch values for them to be of unit variance and zero mean, and, secondly, rescaling these values with an affine transformation of learnable parameters. Denoting as x a sample in the batch, μ and σ respectively the average and standard deviation of the values in this batch, the batch normalization layer applies the transformation:

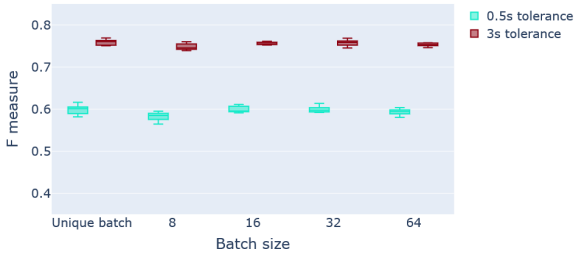
$$\mathrm{bn}(x) = \left(\frac{x-\mu}{\sigma}\right) s + m \tag{B.1}$$

where parameters s and m are trained in the optimization process, and can be seen as new standard deviation and average for the outputs of the batch normalization layer.

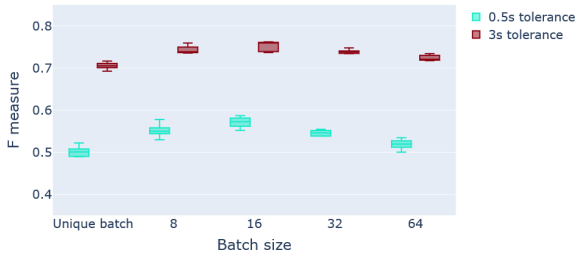
Batch normalization layers often lead to a gain in performance, while speeding up the training process [IS15]. Indeed, by normalizing the input of each layer, the optimization process is less dependent on the distribution of layer's weights, which may vary with the



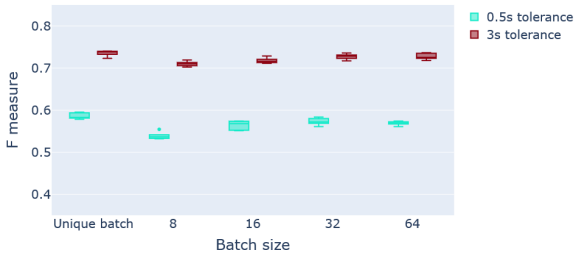
(a) Cosine autosimilarity of Log Mel representation.



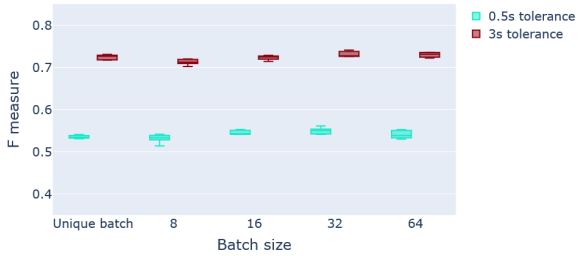
(b) Covariance autosimilarity of Log Mel representation.



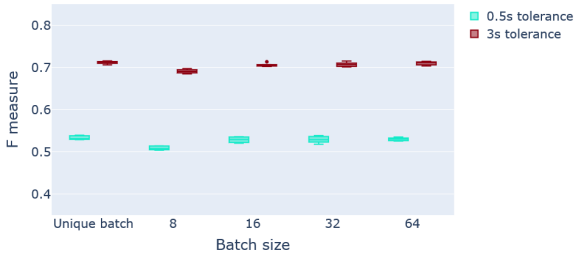
(c) Cosine autosimilarity of NNLM representation.



(d) Covariance autosimilarity of NNLM representation.



(e) Cosine autosimilarity of Mel representation.



(f) Covariance autosimilarity of Mel representation.

Figure B.3 – Segmentation results for different batch sizes, for the FC SSAE, with different features and the Cosine and Covariance autosimilarities (the RBF autosimilarity is not presented, because results were mainly similar to those of Covariance). The different conditions seem to lead to different conclusions, complicating the decision-making process.

different epochs and optimization steps, especially in the first epochs due to the random initialization. Hence, batch normalization layers can stabilize the optimization process with respect to the scaling of weights.

Practically, batch normalization layers are implemented for each hidden layer, before

the input, as presented in Figure B.4 for the FC SSAE. For the Conv network, batch normalization layers are implemented after the pooling layer.

In our particular case, as discussed in previous Section B.2, the song is processed in a unique batch. Hence, here, "batch normalization" actually refers to the normalization of the entire dataset. For the simplicity of notation and convention, we keep the name "batch normalization", even if not accurate for our context.

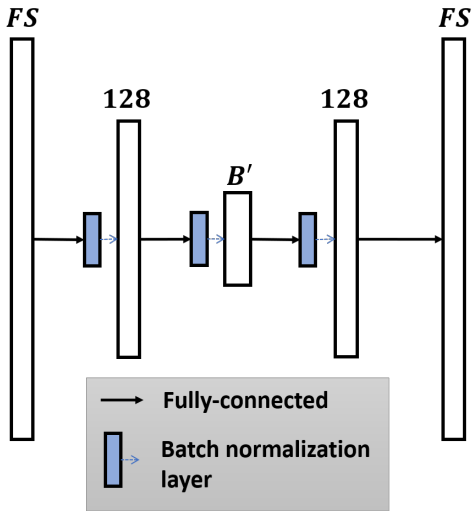


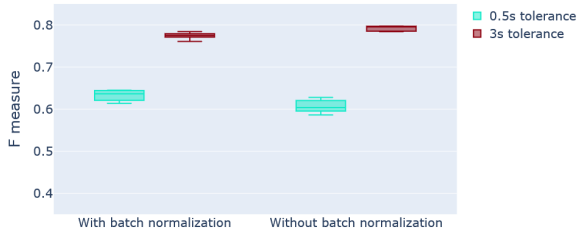
Figure B.4 – Architecture of the FC SSAE with batch normalization layers.

We report results of the Conv and FC networks with and without batch normalization layers on Figures B.5 and B.6 respectively. For both SSAE, the use of batch normalization layers seems beneficial.

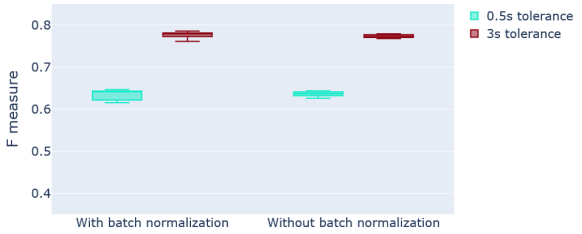
For the Conv SSAE, the use of batch normalization layers generally leads to similar or higher segmentation scores for the Cosine autosimilarity, as presented in Figure B.5 for the Log Mel and MFCC features. These differences are dampened when using another kind of similarity, such as presented with the Covariance autosimilarity in Figure B.5.

For the FC SSAE, the use of batch normalization layers also leads to higher segmentation scores when computing the Cosine autosimilarity. Conversely, on both Covariance and RBF autosimilarities, the FC SSAE obtains better results without batch normalization layers, as presented in Figure B.6.

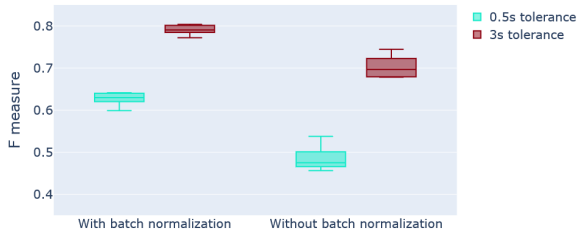
Overall, despite this undecisive conclusion that depends on the similarity function, the variability between the different initializations is lower (*e.g.* a lower inter-quartile difference), which indicates a more robust optimization process. Lowering the variability in results is a motivation for the use of batch normalization layers in our case and in [IS15].



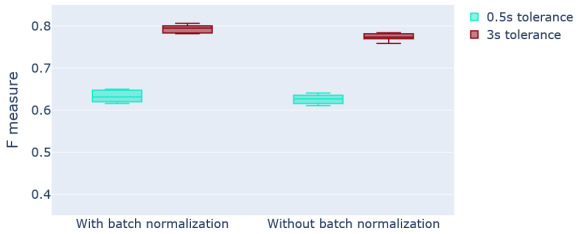
(a) Cosine autosimilarity of Log Mel representation.



(b) Covariance autosimilarity of Log Mel representation.



(c) Cosine autosimilarity of MFCC representation.

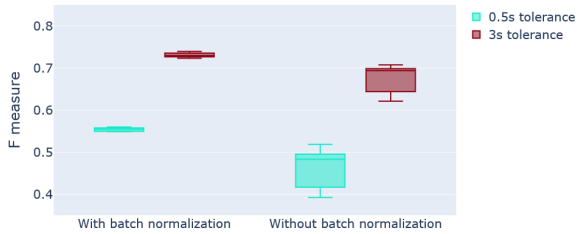


(d) Covariance autosimilarity of MFCC representation.

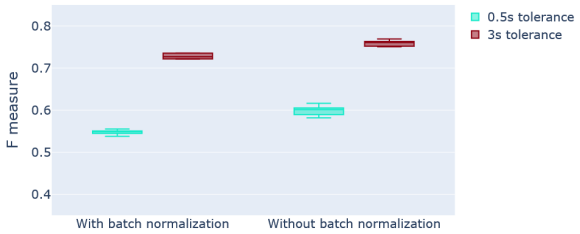
Figure B.5 – Segmentation results for the Conv network, with and without batch normalization layers. Results are presented on the Cosine and Covariance autosimilarities, with different features (namely Log Mel and MFCC).

For both networks, segmentation scores are equivalent between the different autosimilarities when using batch normalization layers, while, without batch normalization layers, results vary according to the type of similarity. This difference in robustness with respect to the similarity may indicate that, with batch normalization layers, SSAEs produce Cosine autosimilarities with relatively higher contrast than without.

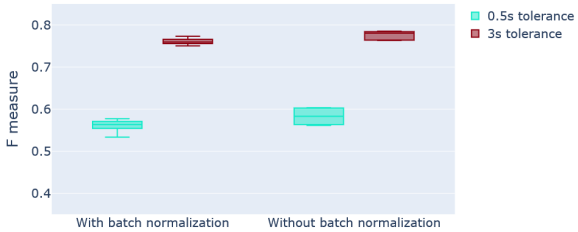
Empirically, this difference in contrast is exhibited in Figure B.7, presenting the Cosine autosimilarity of the Log Mel spectrogram of the song *POP01* of RWC Pop. Even if an example cannot be used as a proof or for generalization, this visual confirmation on an example strengthens the hypothesis of a better contrast with the Cosine autosimilarity when using batch normalization layers, *i.e.* a more accurate notion of dissimilarity between dissimilar bars.



(a) Cosine autosimilarity of Log Mel representation.

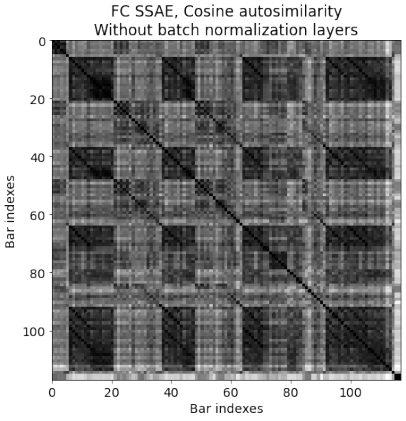


(b) Covariance autosimilarity of Log Mel representation.

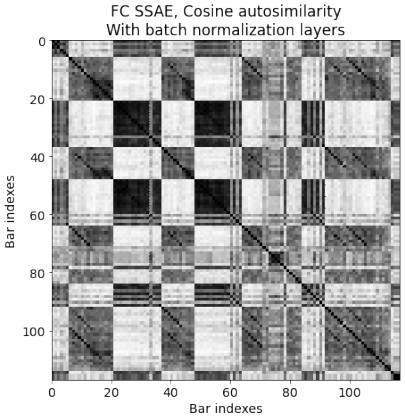


(c) RBF autosimilarity of Log Mel representation.

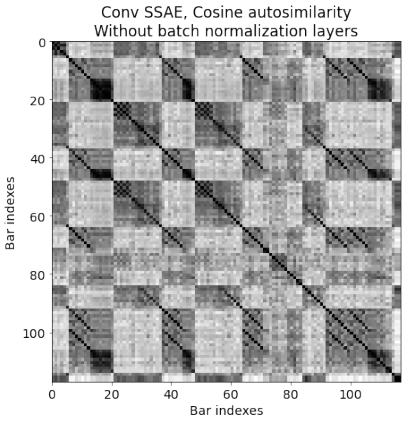
Figure B.6 – Segmentation results for the FC network, with and without batch normalization layers. Results are presented on the Log Mel feature, with different autosimilarities.



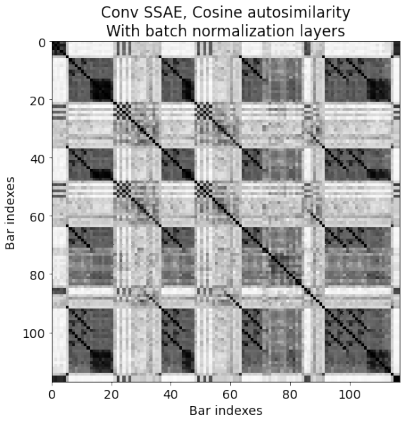
(a) FC SSAE, without batch normalization layers.



(b) FC SSAE, with batch normalization layers.



(c) Conv SSAE, without batch normalization layers.



(d) Conv SSAE, with batch normalization layers.

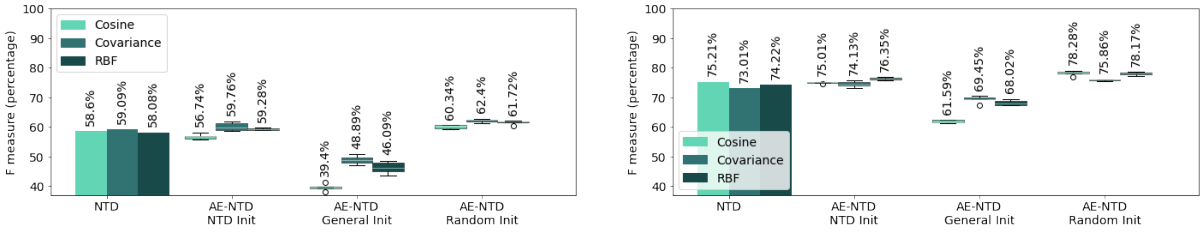
Figure B.7 – Cosine autosimilarities of the different SSAE, with and without batch normalization layers.

Appendix C

General Init for the KL-AE-NTD

Unexpectedly, the General Init condition performs worse than the others, with all similarity functions, while it was expected to obtain (at least) similar performance than those of NTD. We try to explain this result by studying the structural segmentation performance and with an empirical evaluation of the latent embeddings.

As presented in Figure C.1, the Covariance and RBF autosimilarities obtain better results than the Cosine autosimilarity, which probably indicates that the Cosine similarity in this condition is not contrasted enough between similar and dissimilar passages. This is a first hint towards the explanation of this result, *i.e.* that the barwise similarity in the latent space of the General Init KL-AE-NTD is not well suited to the study of structure. Still, both Covariance and RBF autosimilarities perform worse than the Cosine autosimilarities of the other conditions.



(a) $F_{0.5}$ (b) F_3

Figure C.1 – Segmentation results for the KL AE-NTD, depending on the initialization of the decoder, compared with the NTD.

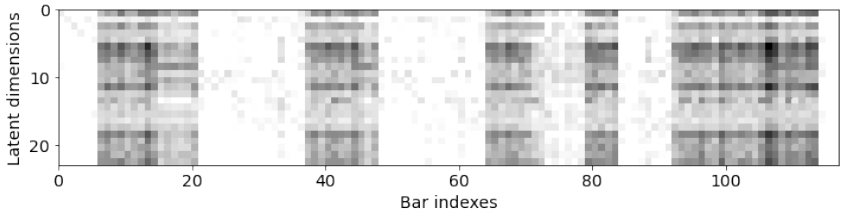
In addition, while, for the Euclidean-AE-NTD, the General Init resulted in a lower reconstruction error than for the NTD (Table 6.1), the opposite trend appears for the KL-AE-NTD: a higher reconstruction error for the General Init than for the NTD, as presented in Table C.1. It may indicate that the optimization paradigm is not relevant, for example because matrices W_{RWC} and H_{RWC} are not suited for the task, or because the optimization paradigm often ends in an irrelevant local minima.

Method		Reconstruction error
NTD		0.41
AE-NTD	NTD Init	**0.36**±0.003
	General Init	0.90±0.011
	Random Init	1.04±0.128

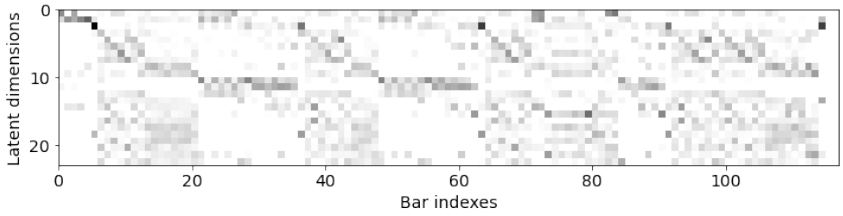
Table C.1 – Reconstruction error, average for a bar in the song, in term of KL-divergence, for the different techniques.

Figure C.2 compares the latent embeddings of the General Init KL-AE-NTD with both the Random Init KL-AE-NTD and the NTD (Q matrix), for the song *POP01*. In this example, for the General Init KL-AE-NTD, all the latent dimensions focus on representing a same group of bars, forming approximately half the number of bars in the song, while the remaining of the song seems neglected. This behavior does not appear for both the Random Init AE-NTD and the NTD, even if the same group of bars seems represented by numerous dimensions in the Random Init AE-NTD (still, not all the dimensions).

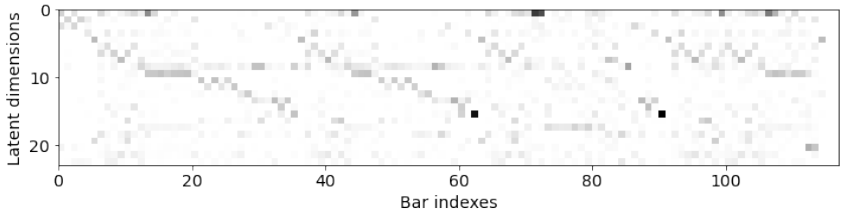
This may explain the poor performance, as the optimization paradigm for the General Init seems to focus on a small number of bars to represent the entire song (at least in this example), and could also explain the high reconstruction error. Still, we do not understand why the optimization paradigm focuses almost exclusively on an excerpt of the song, but we suggest that sparsity constraints could be effective to counteract this effect.



(a) AE-NTD, General Init



(b) AE-NTD, Random Init



(c) KL NTD

Figure C.2 – Q matrices (latent projections for the AE-NTD) of the General Init AE-NTD compared with those of the Random Init AE-NTD and the NTD, computed on the NNLMS of the song *POP01*, for $F' = S' = B' = 24$.

Appendix D

Additional Articles, Not Presented in this Thesis

D.1 Semi-Supervised Convolutive NMF for Automatic Piano Transcription

Presentation of the Article

This article [WMC22] presents the paradigm of Semi-Supervised Convolutive NMF, with application to Music Transcription. This article was published at the SMC22 conference. As this work is independent from the main developments of the PhD work, focusing on structural segmentation of Music, it is not presented in the main part of the manuscript.

SEMI-SUPERVISED CONVOLUTIVE NMF FOR AUTOMATIC PIANO TRANSCRIPTION

Haoran Wu
INSA Rennes, France
haoran.wu@insa-rennes.fr

Axel Marmoret
Univ Rennes, Inria, CNRS,
IRISA, France
axel.marmoret@inria.fr

Jérémy E. Cohen
Univ Lyon, INSA-Lyon, UCBL,
UJM-Saint Etienne, CNRS, Inserm,
CREATIS UMR5220, U1206, France
jeremy.cohen@cnrs.fr

ABSTRACT

Automatic Music Transcription, which consists in transforming an audio recording of a musical performance into symbolic format, remains a difficult Music Information Retrieval task. In this work, which focuses on piano transcription, we propose a semi-supervised approach using low-rank matrix factorization techniques, in particular Convolutive Nonnegative Matrix Factorization. In the semi-supervised setting, only a single recording of each individual notes is required. We show on the MAPS dataset that the proposed semi-supervised CNMF method performs better than state-of-the-art low-rank factorization techniques and a little worse than supervised deep learning state-of-the-art methods, while however suffering from generalization issues.

1. INTRODUCTION

Automatic Music Transcription (AMT) is the task of transforming music recordings into symbolic format, such as scores or MIDI. It is a fundamental musical skill to acquire, taught from early age up to professional level in music schools and, given enough training, humans can be extremely accurate at transcription. Automatic music transcription aims at accelerating and improving time-consuming manual transcription and has applications in music tutoring and rehearsing, musicology analysis or in other music information retrieval tasks [1].

However, while audio generation from MIDI is rather mature, its counterpart AMT is still a very challenging task, even in scenarios involving a single multipitch instrument like a piano, which is our case study. As reported in the 2018 survey by Benetos *et. al.* [1], there are mainly two families of methods to perform AMT: 1) Methods based on low-rank factorizations of spectrograms, and in particular Nonnegative Matrix Factorization (NMF). These methods are mostly unsupervised [2–4]. 2) Deep Neural Networks (DNN) which are heavily supervised. They require registered symbolic-audio training data in a large amount, which can be hard to acquire [5–9].

A recent outbreak in the task of piano transcription (as well as other related tasks) is due to the release of the MAESTRO dataset [7], a large dataset of tightly matched MIDI and audio piano recordings of professional quality which improved the training quality of deep learning techniques. However, the supervised methods require extensive amounts of training data which may not be available for most instruments. The quality of the MAESTRO dataset comes from the existence of the Yamaha Disklavier™, which enables co-recording of audio and MIDI. This high level technology does not exist for most instruments, and building large training dataset for most polyphonic instruments would be extremely challenging on the practical side.

In contrast, since unsupervised factorization-based approaches do not require training data, they obviously solve the data frugality and generalization problems at the cost of being far less accurate than deep supervised approaches.

The goal of this paper is two-fold. On a first hand, leveraging training data available only in limited quantity. On another hand, deploying a variant of NMF, coined Convolutive NMF, in the context of transcription, to improve the transcription performance with respect to NMF. The most closely related work is surely the Attack Decay model [3], which also performs semi-supervision, and proposes a model reminiscent of CNMF. The major differences between the proposed CNMF framework and this work of Cheng *et. al.* are discussed in Section 2.2. Moreover, in Section 4, we show that the performance of the proposed approach are generally much higher and can reach the performance levels observed with Deep Learning at the cost of poor generalization properties. In [4], authors also consider CNMF for piano transcription but CNMF is not the main focus of their work.

This paper is organized as follows: in Section 2, we review the basics of NMF and CNMF for transcription. In Section 3, semi-supervised CNMF is introduced. In Section 4 we show experimental results on MAPS and MAESTRO. Section 5 is devoted to discussions and perspectives.

Notations: Matrices and higher-order arrays are denoted by capital letters, T_{ijk} is the element (i, j, k) in the three-way array T. To denote slices, we use semicolons, so that $T_{i::}$ denotes for instance the slice of all elements of T on row i. Finally, we denote $T_{[a:b]jk}$ elements (i, j, k) with $i \in [a, b]$.

2. CNMF FOR TRANSCRIPTION

2.1 NMF and CNMF Formalisms

Given an element-wise nonnegative matrix $M \in \mathbb{R}_+^{n\times m}$ indexed as M_{ft} with $f \in [1,n],\ t \in [1,m]$, Nonnegative Matrix Factorization (NMF) is a low-rank approximation technique that summarizes M as a sum of rank-one parts, such that

$$M_{ft} = \sum_{q=1}^{r} W_{fq} H_{qt} \tag{1}$$

where $r \leq \min(n,m)$ is a user-defined parameter relating to the number of patterns underlying M, see Figure 1. In practice, when M is an amplitude spectrogram, such as in this work, NMF is computed approximately and boils down to solving a bi-level constrained optimization problem

$$\underset{W\in\mathbb{R}_+^{n\times r}, H\in\mathbb{R}_+^{r\times m}}{\operatorname{argmin}} D_{KL}(M, WH) \tag{2}$$

where $D_{KL}(M,WH)$ is the element-wise Kullback-Leibler divergence between matrix M and its nonnegative low-rank approximation $WH = \sum_{q=1}^{r} W_{:q}H_{q:}$. In AMT, parameter r often relates to the number of notes expected in the recording, and therefore is generally set to (sometimes a multiple of) $r = 88$ for piano recordings [2].

Furthermore, factor matrices W and H are respectively related to pitch and time activation. More specifically, each column of W is expected to contain a spectral template characteristic of a single pitch on the instrument used in the recording, while each corresponding row in H is expected to provide the activation of that note in the recording [10], see Figure 2.

An immediate critic about applying NMF to AMT is that reducing a note to a single frequency template, even tailored for a given instrument, is too restrictive. In practice, frequency templates should evolve with both amplitude and time. While explicit amplitude dependence would break the principle of low-rank approximation underlying NMF, it is possible to extend NMF to include a time-dependence on the templates, which yields Convolutive NMF [12]:

$$\underset{W\in\mathbb{R}_+^{n\times\tau\times r}, H\in\mathbb{R}_+^{r\times m}}{\operatorname{argmin}} D_{KL}(M, \sum_{q=1}^{r} W_{::q} * H_{q:}) \tag{3}$$

where $[W_{::q} * H_{q:}]_{:t} = \sum_{i=0}^{\tau-1} W_{:iq}H_{q(t-i)}$ is a discrete convolution and $q \in [1,r]$, see Figure 1 for an illustration. By convention, we set $H_{q(t-i)} = 0$ whenever $t-i \leq 0$. Integer τ is again a user-defined hyperparameter that dictates the size of the convolution window. To provide a different perspective, the element-wise noiseless CNMF also writes

$$M_{ft} = \sum_{q=1}^{r}\sum_{i=0}^{\tau-1} W_{fiq}H_{q(t-i)}\ . \tag{4}$$

In a nutshell, CNMF enriches NMF by allowing each note to have a full STFT matrix $W_{::q}$ as a frequency template instead of a single column. Therefore, it may also captures time-dependent events such as echoes or non-uniform partials attenuation. It can also be interpreted as a constrained NMF with large rank $r \times \tau$ where each note is represented by τ templates, and the corresponding τ rows in H are constrained to be equal up to a shift. Other works have also considered enriching NMF with several templates per note albeit not using convolution, typically by fusing rows of the estimated H matrix a posteriori [13–15].

2.2 Comparing the Attack Decay Model With CNMF

A reader familiar with the work of Cheng *et. al.* [3] will notice that our work is similar in several aspects with their proposed Attack Decay (AD) framework for music transcription, but let us properly compare the models. After some rewriting of the original AD (see additional material [1]), AD decomposes the data M into two terms

$$M_{ft} = \underbrace{\sum_{q=1}^{r}\sum_{i=0}^{2\tau}\left(\tilde{W}_{fq}^{\text{attack}}P_{-i}\right)H_{q(t-i)}}_{\text{one note attack}} \tag{5}$$

$$+\underbrace{\sum_{q=1}^{r}\sum_{i=\tau}^{t+\tau-1}\left(\tilde{W}_{fq}^{\text{decay}}e^{-\alpha_q(i-\tau)}\right)H_{q(t-i)}}_{\text{one note decay}}\ . \tag{6}$$

It thus appears that the attack term is a CNMF with rank-one templates $W_{fiq}^{(\text{CNMF})} = W_{fq}^{\text{a}}P_i$ which is therefore less general than the CNMF model. The decay term is also a CNMF with rank-one templates.

With some further manipulations, one can see that it is possible to entirely recast the AD model as a CNMF model with rank-two templates, which may explain the performance gap between the two models observed in Section 4. Indeed in the semi-supervised setting, we seem to have enough data to learn unconstrained templates W^{train}, and the Attack-Decay structure on the templates may not be beneficial.

3. TEMPLATE LEARNING AND CNMF

3.1 Challenges in Unsupervised CNMF

In the context of music transcription, it is rarely discussed why NMF performs extremely well on simple dataset, but rather poorly on more complex ones. Saying that NMF, or CNMF, is a part-based representation with no destructive interferences between components does not explain this behavior. In fact, supposing the data indeed is generated reasonably well with a "ground-truth" NMF $M = AB$ for some true frequency templates $A \in \mathbb{R}_+^{n\times r}$ and activations $B \in \mathbb{R}_+^{r\times m}$, we need to ensure that computing an exact NMF $M = WH$ will indeed yield $A = W$ and $B = H$. In other words, the data M must admit a unique NMF.

Theoretically speaking, it is known that NMF will only enjoy this uniqueness property in particular cases, such as when sources are sufficiently scattered or when the data is very sparse [16, 17]. While this may hold for simple songs where notes do not overlap a lot, in the general case one should **not** expect that W and H behave as expected without restricting the set of solutions. Even worse, CNMF

[1] `https://github.com/cohenjer/TransSSCNMF`

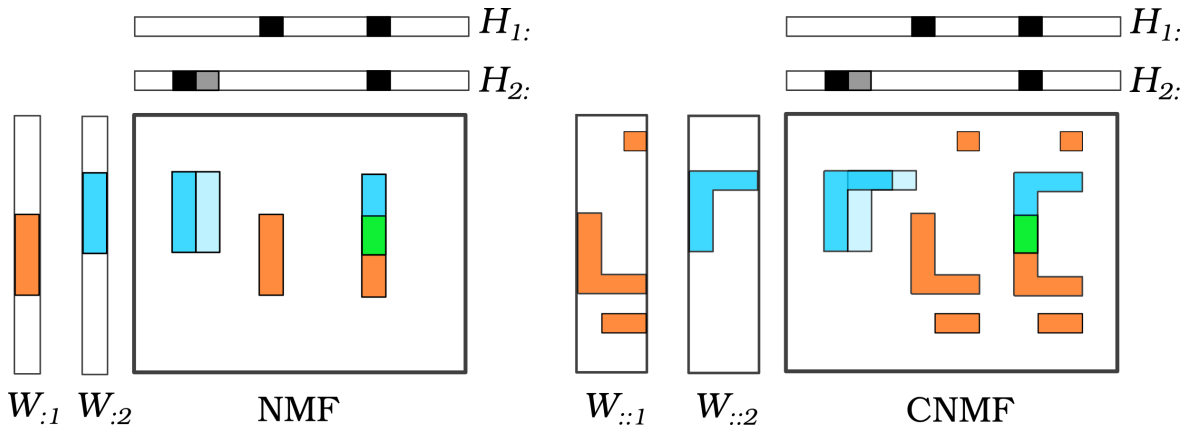


Figure 1. A visual comparison of NMF (left) and CNMF (right). CNMF allows to model complex time dependance while maintaining the number of templates low.

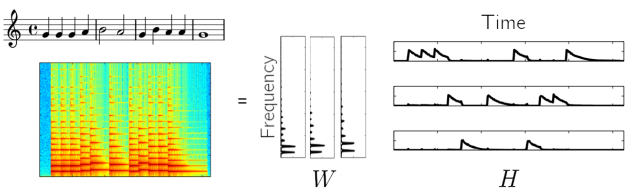


Figure 2. A toy example of transcription using NMF (adapted from [11]).

being a generalized NMF model, it is bound to have even weaker uniqueness properties than NMF (but nothing is known on CNMF identifiability to the best of our knowledge). Blind CNMF has been used with additional sparsity constraints for drums transcription, but dealing with drums typically yields much sparser and lower-rank data than pitched audio due to the temporal localization of percussive sounds.

Therefore, in general, unsupervised CNMF is not regularized enough to perform transcription. While some works focus on further regularization of NMF [18], we instead turn towards semi-supervision.

3.2 Learning Note-Wise Templates

Our working hypothesis is that audio recordings of isolated pitches are available, similarly to what is used for virtual instruments, except that we only make use of one template per note. Each recording is processed as the module of its complex STFT, denoted $V_{::q} \in \mathbb{R}_+^{n \times m_q}$ where m_q is the number of STFT frames for that recording. For a regular piano one needs 88 such templates. Apart from pitch knowledge, no registered MIDI information is required.

The goal of the learning phase here is to estimate $W_{::q}$ for each q using each individual recording $V_{::q}$. We propose to compute an approximate rank-one CNMF of each $V_{::q}$ to estimate $W_{::q}$ and h_q^{train}, the latter being discarded after the training phase. From a theoretical perspective, rank-one CNMF is a constrained version of NMF of rank τ, furthermore computed on a very simple dataset. Therefore it fulfills the qualitative NMF uniqueness criteria discussed above, and we expect the recovered W to contain adequate

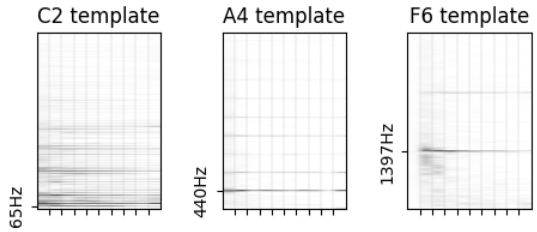


Figure 3. Three trained templates from the AkPnCGdD synthetic piano in MAPS, using $\tau = 10$ convolution size. Templates $W_{::q}$ have been square rooted to better highlight higher frequencies.

note frequency templates.

Practically, we solve for each $q \in [1, r]$ the following optimization problem

$$W_{::q}^{\text{train}}, h_{q:}^{\text{train}} \in \underset{W \in \mathbb{R}_+^{n \times \tau \times 1},\, h \in \mathbb{R}_+^{1 \times m}}{\operatorname{argmin}} D_{KL}(V_q, W * h) \quad (7)$$

using a recently proposed multiplicative algorithm [19] which alternates between W and h updates while preserving nonnegativity and ensuring cost decrease.

In spite of the rank-one approximation and the simple data, the optimization problem still proves challenging with many local minima. Therefore initialization plays an important role in the learning phase. Because it is reasonable to look for $W_{::q}$ in the $V_{::q}$ data itself, we set

$$W_{::q}^{\text{init}} = V_{:[t^*:t^*+\tau-1]q} \text{ and } t^* = \underset{t \leq m_q}{\operatorname{argmax}} \|V_{:[t:t+\tau-1]q}\|_1 \quad (8)$$

which amounts to finding the τ consecutive columns with most energy for initialization. Then we fill $h_{q:}^{\text{init}}$ with zeros and place a one at t^*. Note that this initialization procedure mimics a recently proposed algorithm for separable CNMF [2] [20] but is less computationally intensive. A total of 500 outer iterations are performed to learn a single note template.

Once the training phase is over, for a single multipitch

[2] Separable CNMF is a computationally simpler variant of CNMF which looks for all matrices $W_{::q}$ in the data itself.

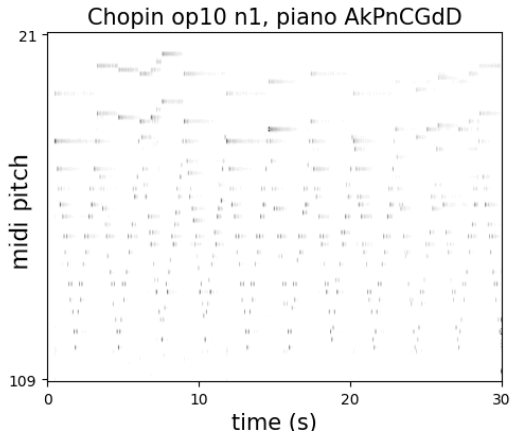


Figure 4. An example of H^{test} computed through rank-one CNMF.

instrument, we have at our disposal the whole dictionary W^{train}, see Figure 3.

3.3 CNMF Transcription With Templates

Testing in the semi-supervised framework only consists of computing the time activations H for a given music excerpt M to transcribe, since W has been pre-trained. This makes the transcription task much easier since the problem

$$H^{\text{test}} \in \underset{H \in \mathbb{R}_+^{r \times m}}{\operatorname{argmin}} D_{KL}(M, \sum_{q=1}^{r} W_{::q}^{\text{train}} * H_{q:}) \qquad (9)$$

is convex and therefore can be solved up to arbitrary precision with the algorithm proposed in [19]. In practice 100 iterations are used, which is generally enough to reach convergence. Initialization was carried out using a few iterations of NMF with W fixed as the first column of each trained template $W_{::q}^{\text{train}}$. An example output H^{test} is provided in Figure 4.

3.4 Post-Processing of Activations

The post-processing of H^{test} that produces a MIDI file matters a lot. Hopefully, prior works have already proposed quite efficient post-processing using an adaptive threshold [3]. We essentially use the same technique but simplified.

In short, activation values in each row of H^{test}, averaged over several consecutive frames, are added to a user-defined threshold δ, defining an adaptative threshold. An onset is detected at the position where the signal is above this adaptive threshold, see Figure 5 for an illustration. Formally, an onset is detected at frame t for note q when

$$h_{qt} > \frac{1}{21} \sum_{j=-10}^{10} h_{q(t+j)} + \delta, \qquad (10)$$

using zero-padding when necessary. The activations are typically very sparse, so we generally did not observe spurious double peaks using the adaptive threshold contrarily to what was observed in [3].

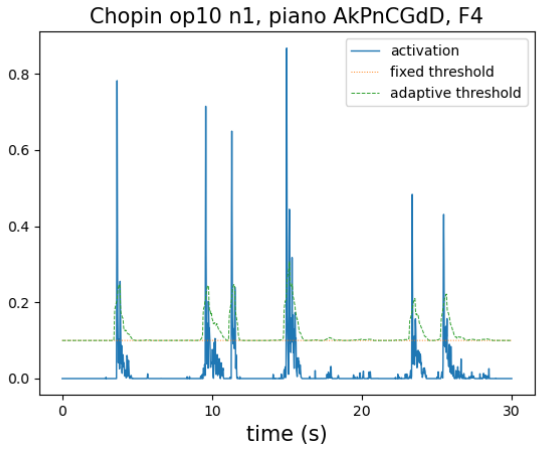


Figure 5. The CNMF activation and adaptive peak-picking method shown for note F4 using a song from MAPS.

4. EXPERIMENTS ON MAPS AND MAESTRO

Although the proposed semi-supervised CNMF framework works in principle for transcribing any multipitch instrument, we only evaluate the performance for piano transcription as a proof on concept. Among the few existing open piano recordings dataset with registered audio and MIDI, in Section 4.2 we focused especially on MAPS [21] which has several kinds of individual notes recordings for several pianos, both virtual and acoustic. We also used MAESTRO [7] to evaluate generalization performance in Section 4.3. In our tests, we only considered the first 30 seconds of each song, as in [3]. Results are discussed in Section 5.

4.1 Experimental Setup

Let us briefly state the various experimental parameters required to reproduce the experiments [3] . All time signals are sampled at 44100Hz, the STFT is computed with windows of 80ms (3528 samples) with a hop-length of 20ms (882 samples). This results in $n = 4097$ frequency bins and $m = 1501$ time frames in the STFT for $30s$ of raw audio signal. No smoothing is applied to the STFT, and we set M as the amplitude spectrogram.

The τ values are chosen among $\tau = 5, 10, 20$. We used one template for each piano note such that $r = 88$. Finally to fix the peak-picking threshold for H^{test}, two oracle strategies are used: 1) use the same threshold for all songs, and report results for the best value on the grid $[0.01 : 0.01 : 0.4]$ 2) perform transcription with a song-dependent threshold, optimized on the same grid. The first case corresponds to a scenario where the threshold is pre-trained for a category of recording (music genre, recording conditions) while the second case corresponds to a hand-tuned threshold for a specific song to transcribe.

We compare our method with the Attack Decay (AD) model presented in Section 3 which, to the best of our knowledge, is the current state-of-the-art for

[3] Python code to compute CNMF and reproduce all the experiments is available at `https://github.com/cohenjer/TransSSCNMF`

unsupervised/semi-supervised piano transcription. The results reported in Table 1 are the exact results from [3] (AD [3]), and the results of AD when applying our post-processing (AD*). In both cases transcription is performed on H^{attack} as defined in [3]. Despite our efforts we were unable to exactly reproduce the original AD scores. In particular the original AD paper introduces smoothness in several aspects: the data spectrograms are locally averaged, and peak-fusion is performed in the post-processing. Consequently, the AD* results enable comparison between the proposed CNMF and AD in the same pre/postprocessing conditions, while AD [3] are the best results achieved by Cheng *et. al.*. For completeness, we also report the transcription score from the state-of-the-art piano transcription network introduced in [8] which was trained on MAESTRO [7].

To measure performance, we compute a notewise score using the mir_eval [22] toolbox with a tolerance of 50ms. The offset detection problem is not tackled. Results are shown using only F-measure (F) and Accuracy (A) metrics (reported in percent), but full results including Precision and Recalls for all pianos are available in the complementary materials online.

4.2 Transcription Performance on MAPS

The MAPS dataset contains classical piano music pieces recorded with different pianos and conditions: a Yamaha Disklavier™in two settings 'ENSTDkCl' (EN1) and 'ENSTDkAm' (EN2), and six synthetic pianos 'AkPnBcht', 'AkPnBsdf' (AkB1-2), 'AkPnCGdD' (AkC), 'AkPnStgb' (AkS), 'SptkBGAm' (Sp), 'StbgTGd2' (St). For each piano/setting listed above, we train a template W^{train} using a rank-one CNMF as presented in Equation 7. Since there are many available single notes recordings in MAPS, we chose based on performance to use the Isolated notes (ISOL / NO) recorded at Medium intensity (M).

In a first experiment, we study the sensitivity of the proposed method to the selection of the convolution window size τ and the choice of a threshold δ tuned on the whole corpus versus on each song individually. We also compare our results to the Attack Decay model and the ByteDance supervised neural network. Results are shown in Table 1. From this experiment, we see that generally $\tau = 10$ performs best, and that the song-wise threshold gives better results.

In a second experiment, the templates for all other pianos are used to transcribe AkPnCGdD and ENSTDkCl to estimate the generalization capacities of the trained CNMF templates, see results in Table 2. Only CNMF with song-tuned threshold is shown and we set $\tau = 10$, to show the best results only.

Additionally, Table 3 reports the average running times when training the templates and performing transcription on the AkPnCGdD recordings. This test was run on a personal computer with AMD Ryzen 5 2600™ processor and 16GB RAM.

4.3 Generalization on MAESTRO

A natural question regarding CNMF templates is how well they can be used outside their training context without any domain adaptation. While results shown in Table 2 already provide a partial answer, we also tried to apply CNMF to the MAESTRO dataset. However, since no individual notes recordings are publicly available for MAESTRO, we used the templates learnt from MAPS. We transcribed 20 songs from the MAESTRO test set randomly chosen.

The results are quite poor: even when choosing song-wise thresholds, for all templates, CNMF does not reach above 59% in F-measure (test results are available in the supplementary materials). For comparison, the state-of-the-art with supervised deep learning techniques reaches above 95% F-measure on MAESTRO. Its performance on MAPS with data augmentation are also state-of-the-art, around 89% F-measure on EN1, despite the training/testing mismatch.

5. DISCUSSION

In light of the experiments conducted in Section 4, let us discuss the strengths of the proposed CNMF. It exhibits a significant improvement with respect to the Attack Decay model, which as far as we know is state-of-the-art for semi-supervised piano transcription. This is even more true when using the same pre-processing and post-processing for AD and CNMF, the former being in particular prone to unstable activations which were not observed in the latter. We may therefore affirm that the improvement in performance is indeed due to the CNMF model design. In other words, CNMF with a semi-supervised setting is an efficient piano transcription method. From numerical results, it seems that a convolution window size $\tau = 10$ is a good compromise between quality of transcription and transcription computation time.

The CNMF method does not perform better than the supervised state-of-the-art method we denoted as ByteDance DNN, which is expected given that this neural-network competitor is trained on MAESTRO which contains more than two hundred hours of perfectly aligned MIDI and audio piano recording of professional level. We still reach similar performances on some pianos such as EN1, EN2 and AkS. Nevertheless, the ByteDance DNN is not trained on MAPS contrarily to the proposed semi-supervised CNMF.

Moreover, the proposed semi-supervised setting only requires a handful of training dataset which are relatively easy to acquire. Indeed, only individual notes recordings are necessary, without any audio and MIDI registration. Compared to the very large amount of data currently required by state-of-the-art deep learning approaches, this is a huge advantage of the proposed approach applicable to any acoustic instrument with well-defined onsets readily available. Sadly our study is limited to piano transcription. A perspective of this work is to apply it to transcribe polyphonic instruments for which recording registered MIDI and audio is challenging.

Finally, while the performance does depend on the choice

			EN1		EN2		AkB1		AkB2		AkC		AkS		Sp		St	
thresh		τ	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A
global	CNMF	5	78	65	70	55	88	80	75	62	83	72	80	69	81	70	75	61
		10	**85**	**75**	**77**	**64**	93	88	87	78	91	84	**88**	**79**	89	82	84	74
		20	83	72	76	63	**94**	**89**	**87**	**79**	**92**	**86**	87	79	**90**	**83**	**86**	**77**
	AD*		81	69	68	53	66	50	71	56	60	43	67	51	64	47	67	50
song	CNMF	5	82	70	74	59	90	82	80	69	87	78	84	74	86	77	81	69
		10	**88**	**79**	**80**	**68**	**95**	**91**	**90**	**83**	94	89	**90**	**82**	**93**	**87**	89	80
		20	85	75	78	66	**95**	**91**	**90**	**83**	**94**	**90**	89	81	92	87	**89**	**81**
	AD*		82	70	69	54	68	52	73	59	61	45	69	54	66	50	70	54
AD [3]			82	70	-	-	-	-	-	-	85	74	-	-	-	-	-	-
ByteDance DNN [8]			**89**	**81**	77	65	**98**	**97**	**95**	**90**	**98**	**96**	87	77	**97**	**95**	**95**	**90**

Table 1. CNMF, AD, and the ByteDance supervised network performance with respect to the choice of hyperparameter τ and the choice of the peak-picking threshold, without training/testing mismatch for CNMF and AD. Only the first 30s of each songs were used. AD* uses the same pre/post-processing as CNMF. Tolerance is 50ms.

		EN2	AkB1	AkB2	AkS	Sp	St
AkC	F	74	77	77	70	74	77
	A	59	64	63	56	59	63
EN1	F	76	67	68	69	67	69
	A	62	50	52	53	52	53

Table 2. Transcription scores for CNMF, with training/testing mismatch.

	Training			Transcription
τ	5	10	20	10
Av. time	56s	193s	634s	239s

Table 3. Average computation time for a learning pattern (Training) or transcribing 30s of a song (Transcription) for semi-supervised CNMF. Results are reported for the AkPnCGdD piano in MAPS.

of a good activation threshold, CNMF still performs well using a global threshold over all songs in MAPS for each piano. Therefore extensively tuning the threshold hyperparameter is not essential to the success of CNMF here.

Despite these encouraging results, CNMF has a few issues which open interesting perspectives. First, it clearly has a significant generalization problem, or in other words, the learning stage overfits the training data. From Table 2, it appears that a mismatch between training and testing inside MAPS, while detrimental to transcription performance, is not as severe as a learning on MAPS and testing on MAESTRO. A tentative explanation is that the MAESTRO recordings are live performances with quite loud reverberation, while the MAPS recordings are drier. Looking for an audio transformation of the templates that minimizes recording conditions mismatch would therefore probably prove beneficial to generalize pre-recorded CNMF templates. Retraining a template library given few annotated data in the testing set could also be a possible solution. Whether this domain adaptation can be done fully blindly is still unclear however.

Second, despite performance not relying too much on the threshold level, the threshold selection method on the other hand is extremely important. Using a fixed threshold instead of the adaptive peak-picking drastically decreased performances in our early tests. But this also means that the post-processing of activations can be further improved using more involved technique than thresholding each note individually, and this research direction should not be overlooked if transcription performances of CNMF are to be further improved.

Third, for simplicity only one template for each note was used for the transcription phase. However, most instruments sound quite differently depending on how they are played. The proposed semi-supervised framework currently does not account for this timbre variation with amplitude or technique, and adapting the current method to make use of several templates per notes is an interesting research direction.

Finally according to the results shown in Table 3, computation time is rather large even in the testing phase. With the current implementation, real-time processing is therefore prohibited. Using a CNMF solver dedicated to Kullback-Leibler divergence or working on a more efficient rank-one CNMF solver than [19] could nevertheless drastically reduce computation time.

6. CONCLUSION

The state-of-the-art for automatic piano transcription is undeniably nowadays detained by deep learning techniques. However these methods rely on very large audio and symbolic registered dataset which are potentially very hard to obtain. In this work, we propose a competitive semi-supervised matrix factorization model which only requires labeled recordings of each individual notes. We show that when there is no mismatch between the training data and the test data, our approach performs significantly better than semi-supervised state-of-the-art approaches, approaching supervised deep learning performance. Therefore, we believe that using CNMF instead of NMF is an important step towards learning more reasonable frequency templates in low-rank approximation techniques for piano transcription or other similar tasks. Further works should however be devoted to adapt pre-trained templates to reduce generalization error. Improving the onset detection

method, allowing timbre variation in templates and reducing computation time are other important research directions. Finally, the proposed semi-supervised approach should be tested with other instruments than the piano and in a multi-instrument setup.

Acknowledgments

Jeremy E. Cohen and Axel Marmoret thank ANR JCJC LoRAiA ANR-20-CE23-0010 for supporting this work.

7. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.

[2] E. Vincent, N. Bertin, and R. Badeau, "Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 109–112.

[3] T. Cheng, M. Mauch, E. Benetos, and S. Dixon, "An attack/decay model for piano transcription," in *ISMIR 2016-17st International Society for Music Information Retrieval*, 2016.

[4] L. Gao, L. Su, Y.-H. Yang, and T. Lee, "Polyphonic piano note transcription with non-negative matrix factorization of differential spectrogram," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 291–295.

[5] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[6] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[7] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019.

[8] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onsets and offsets times," *arXiv preprint arXiv:2010.01815*, 2020.

[9] Y. Yan, F. Cwitkowitz, and Z. Duan, "Skipping the frame-level: Event-based piano transcription with neural semi-crfs," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[10] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2003, pp. 177–180.

[11] N. Bertin, "Les factorisations en matrices non-négatives : approches contraintes et probabilistes, application à la transcription automatique de musique polyphonique," Ph.D. dissertation, 2009. [Online]. Available: http://www.theses.fr/2009ENST0051

[12] P. Smaragdis, "Convolutive speech bases and their application to supervised speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 1–12, 2006.

[13] T.-M. Wang, P.-Y. Tsai, and A. W. Su, "Score-informed pitch-wise alignment using score-driven non-negative matrix factorization," in *2012 International Conference on Audio, Language and Image Processing*. IEEE, 2012, pp. 206–211.

[14] E. Benetos, A. Klapuri, and S. Dixon, "Score-informed transcription for automatic piano tutoring," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. IEEE, 2012, pp. 2153–2157.

[15] D. Jeong and J. Nam, "Note intensity estimation of piano recordings by score-informed nmf," in *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.

[16] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *In Advances in Neural Information Processing 16*, 2003.

[17] X. Fu, K. Huang, and N. D. Sidiropoulos, "On identifiability of nonnegative matrix factorization," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 328–332, 2018.

[18] V. Leplat, A. M. Ang, and N. Gillis, "Minimum-volume rank-deficient nonnegative matrix factorizations," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3402–3406.

[19] D. Fagot, H. Wendt, C. Févotte, and P. Smaragdis, "Majorization-minimization algorithms for convolutive NMF with the beta-divergence," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8202–8206.

[20] A. Degleris and N. Gillis, "A provably correct and robust algorithm for convolutive nonnegative matrix factorization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2499–2512, 2020.

[21] V. Emiya, N. Bertin, B. David, and R. Badeau, "MAPS-a piano database for multipitch estimation and automatic transcription of music," 2010.

[22] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

D.2 Polytopic Analysis of Music

Presentation of the Article

This article presents the Polytopic Analysis of Music paradigm, and particularly the work of Guichaoua [Gui17]. The *MusicOnPolytopes* toolbox [MCB21] is associated with this article, and both the article and the toolbox were developed at the same time. It should be noted that the toolbox also includes the work of Louboutin [LB17], while not presented in the article. Finally, this article has not been fully reviewed, and is still a working document, which is why it is not presented in the main part of the manuscript.

Polytopic Analysis of Music

Marmoret Axel[1], Jérémy E. Cohen[1], and Frédéric Bimbot[1]

[1]Univ Rennes, Inria, CNRS, IRISA, France.

2021

Abstract

Structural segmentation of music refers to the task of finding a symbolic representation of the organisation of a song, reducing the musical flow to a partition of non-overlapping segments. Under this definition, the musical structure may not be unique, and may even be ambiguous. One way to resolve that ambiguity is to see this task as a compression process, and to consider the musical structure as the optimization of a given compression criteria.

In that viewpoint, C. Guichaoua [1] developed a compression-driven model for retrieving the musical structure, based on the "System and Contrast" model [2], and on polytopes, which are extension of n-hypercubes. We present this model, which we call "polytopic analysis of music", along with a new open-source dedicated toolbox called *MusicOnPolytopes*[1] (in Python). This model is also extended to the use of the Tonnetz as a relation system. Structural segmentation experiments are conducted on the RWC Pop dataset [3]. Results show improvements compared to the previous ones, presented in [1].

1 Introduction

Structural segmentation of music is an important task in the Music Information Retrieval (MIR) community. This task aims at representing musical information at a mesoscopic scale with symbolic information, such as letters or semantic information (verse, chorus, etc). The musical content is hence partitioned and organized into a list of segments. Relevant structural segments must be computed from low-level musical information, thus necessitating the definition of salient metrics to form and evaluate potential segments.

This work presents a compression-based scheme for structural segmentation of symbolic music (i.e. music discretized both in time and representation as a flow of symbols) called "polytopic analysis of music", and introduces an open-source toolbox dedicated to this scheme [4].

The idea of linking music structure and compression schemes probably trace back to works such as Meyer' principles [5], Lerdahl & Jackendoff Generative Theory of Tonal Music [6] and Narmour's Implication-Realization model [7]. These works focus on music perception to capture some sense of music coherence in pieces, and are in that sense knowledge-based models.

On the other hand, probabilistic and Information-Theory-oriented models (hence, models which are less driven by prior knowledge) have been studied to capture structures of songs, such as the IDyOM (Information Dynamics Of Music) which studies the Information Content of musical events [8], models based on the Kolmogorov complexity [9], or, more recently, Stochastic Neural Networks such as Restricted Boltzmann Machines [10].

An exhaustive list of compression-based structural segmentation models is beyond the scope of this article, and such a review can be found in [8]. For its part, this article focuses on the S&C model [2], stemming from Narmour's theory [7].

This work is principally based on the previous work of C. Guichaoua [1], which was only presented in french until now. It is based on geometrical objects, called "polytopes", which support atomic musical elements on its vertices, and allows to study these musical elements in a non-chronological manner.

[1]https://gitlab.inria.fr/amarmore/musiconpolytopes

In that sense, polytopes can highlight repetition in music which don't occur sequentially, and aims at evaluating the information (or, more informally, the "novelty") brought by each musical element. Polytopes are well suited for the compression of musical information, as demonstrated in [11], which shares this Information-Theory point of view to study repetitions and anticipations in music[2].

Polytopic analysis of music focuses on retrieving the frontiers between segments (*i.e.*, structural boundaries), which are the time instances separating two consecutive parts. It does not study the labelling stage of segments, which consists of labelling in a same way coherent segments, and distinguishing dissimilar ones. A complete description of the structural segmentation task can be found in [12].

This article presents a compression-base definition of the structural segmentation task, in Section 2. Then, the main components of polytopic analysis of music are introduced in Section 3, and the compression-oriented cost function associated with these objects is presented in Section 4. Finally, numerical experiments on the RWC Pop database are presented in Section 5.

2 Structural Segmentation as a Compression Scheme

2.1 Definition of the Problem

The polytopic analysis of music considers that the structure in music can be found by evaluating its internal repetitions, and by regrouping the similar passages in sections. Formally, this can be obtained in an optimization scheme, by defining the optimal structure as the structure of maximal compression, *i.e.* the structure minimizing a complexity cost, left to be defined.

This work only considers the compression of music in a symbolic form, meaning that music is discretized both in time and features as a flow of symbols.

Practically, music is considered as discretized on musical beats[3], and symbols represent the 24 major or minor perfect chords, summing up the musical content to the leading triad in the harmony. Let us denote $\{a_t,\, 1 \leq t \leq T\}$ this symbolic representation, a_t being the symbol used to represent music at time t (aligned with beats), and T the length of the song.

The structural task now consists in finding a set of segments $Z = \{S_n,\, 1 \leq n \leq N\}$. Each S_n is a sequence of consecutive elements a_t, such that $S_n = \{a_{t_n}, a_{t_n+1}, ..., a_{t_{n+1}-1}\}$. The set $\{S_n\}$ partitions the $\{a_t\}$, in the sense that every a_t belongs to one and only one S_n. Indexes $\{t_n\}$ are frontiers between segments.

Let's suppose for now the existence of a complexity cost function $\mathcal{C}$ applying on musical passages. The structural segmentation task is now defined as the search of the optimal sequence of segments regarding this cost, *i.e.*, denoting as $\mathcal{C}(S_n)$ the cost of segment S_n:

$$Z^* = \arg\min_{Z} \sum_{n=1}^{N} \mathcal{C}(S_n) \tag{1}$$

This is an optimization problem, which can be solved by a combinatorial analysis of the possible solutions. In particular, Sargent *and al.* [13] presented a dynamic programming algorithm which iteratively computes the optimal segmentation with respect to a cost.

The structural segmentation task is now reframed as the search of a complexity cost function $\mathcal{C}$ for segments, representing the main content of this article.

2.2 Relation Between Musical Elements

The development of this complexity cost function $\mathcal{C}$ is based on the study of relations between elements. Let's denote by A the set of possible elements: $\forall 1 \leq i \leq T, a_i \in A$. In this work, A represents all major and minor perfect chords, *i.e.* $A = [\![0, 23]\!]$.

[2]As a matter of fact, the toolbox *MusicOnPolytopes* also includes the work presented in [11], and extends it for the task of structural segmentation by defining costs for irregular polytopes. Still, as both paradigms differ in numerous points, and for clarity, it is not presented here. An interesting reader should refer to [11].

[3]Other discretization could be considered, but musical beats has the advantage of being a musically-motivated discretization of time.

These elements are studied relatively with the others (and not individually). Precisely, let G_r an abelian group of elements called relations. This group allows us to operate on A, which means that relations in G_r act on the group A.

Hence:

- $\forall f \in G_r$, $\forall a \in A$, $f.a \in A$,
- for all pair of elements $(a_i, a_j) \in A$, $\exists f \in G_r / f.a_i = a_j$.

To simplify notations, we denote as $f_{i/j}$ the relation between a_i and a_j. This group ensures that, for every musical passage $\{a_1, a_2, ..., a_n\}$, one can represent the relation between any two elements of this passage.

2.2.1 Triad Circle

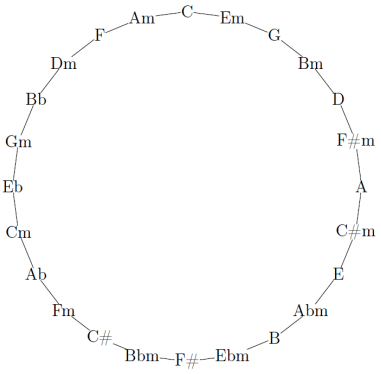
A first set of relations, defined in [1], is the "triad circle". This circle is represented in Figure 1a. Chords in this circle are ordered such that a clockwise rotation of one step represents the increase of the root by a third (*i.e.* using respectively the third and the fifth of the first chord as the root and the third of the second chord).

The relation f between 2 musical elements is defined as the number of steps between these two elements in the circle, or, differently said, the clockwise oriented angle between two elements. Hence, $\forall f \in G_f, f \in [\![-11, 12]\!]$.

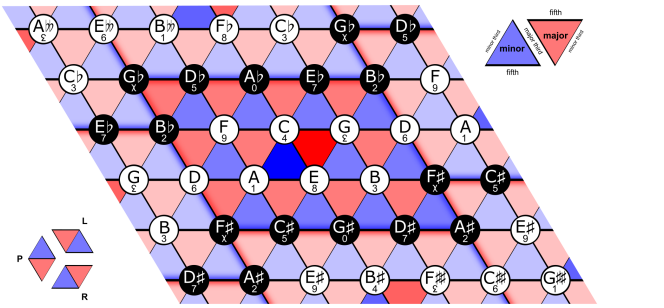
2.2.2 Tonnetz

A second set of relations is based on the Tonnetz, and more particularly the Neo-Riemannian Tonnetz subject to the western 12-chromatic scale [14]. The tonnetz is a lattice whose elements are ordered according to 3 harmonic relations between triads: P (Parallel, for triads sharing a common fifth, which represents here relations between a minor and a major triad sharing the same root, like A major and A minor), R (Relative, for triads sharing a common major third, such as A minor and C major) and L (for Leading-tone exchange, meaning that both triads share a common minor third, such as C major and E minor). It is represented in Figure 1b.

In this Tonnetz, two perfect chords can be compared as a sequence of composition of these three PLR relations. Even if the relation between two chords is not unique, one can define the "canonic" relation between them as the shortest relation in number of PLR relations.[4]



(a) Triad circle (from [1]).



(b) Tonnetz
(By Watchduck (a.k.a. Tilman Piesk) - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=33932849)

Figure 1: Two models of relations: the triad circle and the Tonnetz.

[4]Note that using the tonnetz with only the L and R relations redefines the previous triad circle.

3 Polytopic Analysis of Music

3.1 Polytopes

A polytope is a geometrical pattern, composed of vertices and oriented edges (arrows). Polytopes are defined to scale up the previously defined relations to musical passages. Vertices and arrows of a polytope respectively represent musical elements a_t and their relations f.

Definition 3.1 (Regular polytope)**.** Primary polytopes are n-dimensional hypercubes. They are of the form of a square, a cube, a tesseract, etc.

A n-dimensional regular polytope is defined by its **dimension**: a regular n-dimensional polytope contains 2^n elements. Hence, a 2-dimensional regular polytope represents a square and contains 4 elements; a 3-dimensional regular polytope contains 8 elements and represents a cube; etc. A 3-dimensional regular polytope (called 3-polytope for simplification) is represented in Figure 2.

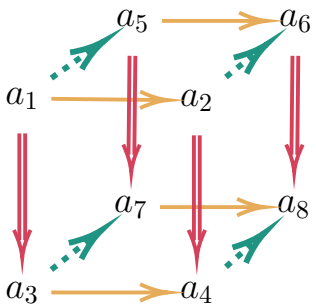


Figure 2: 3-polytope (cube)

Hence, regular polytopes can model musical passages of 2^n elements, but are not suited for passages of different sizes. In order to consider a large number of passages size, we extend these regular polytopes to "irregular polytopes".

Definition 3.2 (Irregular polytope)**.** A n-dimensional irregular polytope correspond to a n-dimensional regular polytope on which has been deleted and/or added some vertices and edges. These alteration (either addition or deletion) follow themselves the shape of a regular polytope of dimension $d < n - 1$, *i.e.* deleted and/or added vertices form themselves a d-dimensional regular polytope.

An irregular polytope is constructed from at most one d-polytope modeling the addition and at most d'-polytope representing the deletion.

For example, starting from a 3-dimensional regular polytope (a cube), and deleting its last vertex (0-polytope) results in a 3-dimensional irregular polytope with 7 elements instead of 8. To detail the construction specifications of these irregular polytopes, we further introduce the notions of **antecedent** and **successor**.

Definition 3.3 (Antecedent)**.** Let an element be the extremity of (at least) one arrow. We define the antecedent(s) of this element as the origin(s) of this (or these) arrow(s). An element can have several antecedents if it is the extremity of several arrows. In Figure 2, a_2 and a_3 are two antecedents of a_4.

Definition 3.4 (Successor)**.** Let an element be the origin of (at least) one arrow. We define the successors of this element as all elements which are at the extremity of this (or these) arrow(s). Elements do not necessarily have successors. In Figure 2, a_4 and a_7 are successors of a_3.

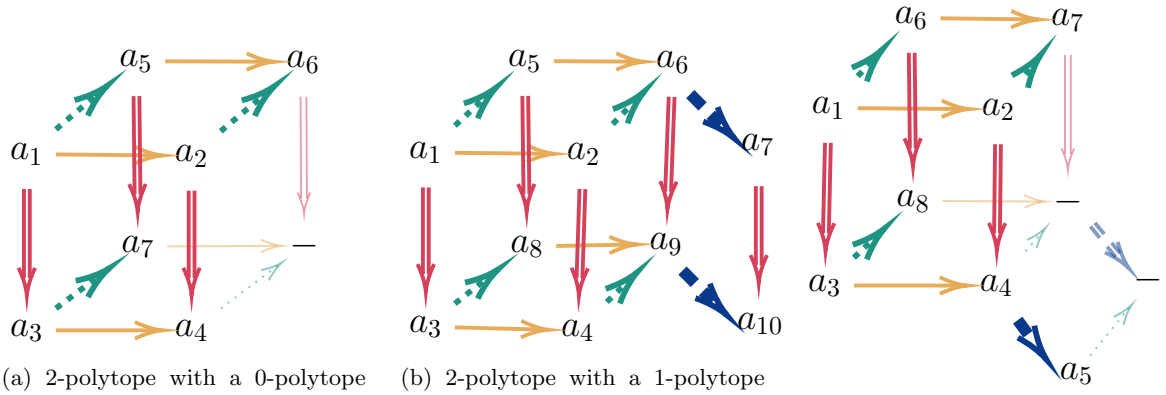
As an edge represents a relation between two elements, every edge must connect existing elements. Hence, deleting a vertex implies the deletion of all arrows starting from its antecedents.

In addition, deleting an element at the origin of an edge implies the deletion of this edge, which can result in elements without arrows connecting them to the polytope. Hence, deleting an element must imply the deletion of its successors.

To ensure this latter condition, every alteration polytope must include the last element of the polytope, and, when both addition and deletion operate on a same vertex, this vertex is deleted without addition (*i.e.* deletion is preferred over addition).

Similarly, every added element must be connected to another element of the polytope by an edge. In that sense, when adding an element, an edge is created with the vertex supporting this addition. This new element is considered as "attached" to this vertex. As the additional edges form themselves a polytope, added elements are connected by new edges.

3 irregular polytopes, respectively with deletion, addition and both, are shown in Figure 3. These polytopes were introduced in [1], and more details are to be found in this work.



(a) 2-polytope with a 0-polytope deletion.

(b) 2-polytope with a 1-polytope addition.

(c) 2-polytope with a 0-polytope deletion and a 1-polytope addition.

Figure 3: 3 irregular polytopes.

3.2 System and Contrast (S&C)

The core of the polytopic analysis lies in the fact that edges between elements model their relations. In that viewpoint, edges can link two elements which are not consecutive in the chronological order, and, hence, model non-sequential relations. This viewpoint is exploited in order to try to anticipate some relations.

Anticipation follows the "System and Contrast" (S&C) model, developed by Bimbot and al. [2]. The S&C model considers a passage of 4 elements, and, by studying the relations between the first 3 elements, tries to anticipate a "fictive" fourth element, compared with the real one.

Formally, when studying $f_{1/2}$, the relation between a_1 and a_2, and $f_{1/3}$, the relation between a_1 and a_3, the 3 elements are now represented by an element (a_1) and two relations ($f_{1/2}$ and $f_{1/3}$). Then, by composing $f_{1/2}$ and $f_{1/3}$, this model defines a fictive fourth element $\hat{a_4}$, implied by the first 3 elements, as $\hat{a_4} = f_{1/2}f_{1/3}.a_1$. The actual fourth element a_4 is then compared to this fictive one, which defines a "contrast" relation γ as $\gamma_{\hat{4}/4}$.

When the fourth element a_4 is equal to the fictive fourth element, a_4 can be deduced from the first 3 elements, thus reducing the amount of information necessary to model 4 elements to 1 element (the first) and two relations. Otherwise, the fourth element is modeled with the contrast relation.

4 Polytopic Complexity Cost of a Musical Passage

Now, starting with polytopes and with the S&C model, this section defines a polytopic complexity cost $C(S, P)$ for a musical passage $S = \{a_1, a_2, ..., a_n\}$ on a polytope P. The polytopic cost is first defined as the sum of the individual costs of each element. Let us start with two useful definitions.

Definition 4.1 (Primer)**.** The first element in the polytope (and in the passage) is called "primer". The

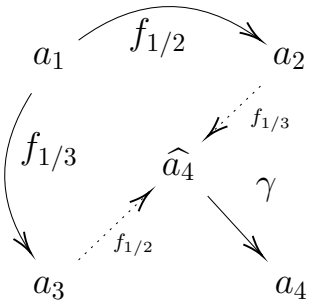


Figure 4: System & Contrast model.

primer does not have any antecedent[5].

Definition 4.2 (Under-primer)**.** Elements whose only antecedent is the primer are called "under-primers". Hence, under-primers are the only successors of the primer.

4.1 Information-Theory-like Viewpoint

In polytopic analysis of music, an element is studied in comparison with previous elements (not necessarily consecutive in chronological order). The cost of an element a_i is denoted as $C(a_i|\{a_1, a_2, ..., a_{i-1}\}, P)$.

This viewpoint aims at finding economical representations of a musical passage, where elements are only encoded if they can't be described by previous elements. This idea is close to the Minimum Description Length paradigm (MDL), an Information-Theory point of view where the shortest description (in terms of quantity of information) is considered the best one.

Here, elements are represented by their relations rather than being entirely described, and some of these relations are anticipated within the S&C model.

All relations belonging to a same group G_r, they can be encoded by a same quantity of information q (for example, representing the number of steps in the circle of triads between two elements leads to a set of 24 relations, requiring an encoding with 5 bits to be entirely described). This quantity of information could be influenced by priors over the distribution of relations or by expert knowledge, but we do not explore that lead in this work. We can further simplify the model by considering that $q = 1$.

Concretely, this means that the complexity cost of a relation is 0 if the relation is the identity, or 1 otherwise. In addition, a_1 can't be described by previous elements, so $C(a_1|P) = 1$. Finally,

$$C(S, P) = C(a_1) + \sum_{i=2}^{n} C(a_i|\{a_1, a_2, ..., a_{i-1}\}, P) \tag{2}$$

with $C(a_i|\{a_1, a_2, ..., a_{i-1}\}, P) \in \{0, 1\}, \forall 2 \leq i \leq n$.

4.2 2-Polytope (Square, 4 elements)

The core of the implication system lies on 4 elements polytopes (which are squares). A square polytope is composed of a primer a_1, two under-primers a_2 and a_3, and a fourth element a_4, which has both under-primers as antecedents. This polytope is evaluated as a S&C model.

The primer must be encoded (as initialization of the passage), so the initial cost of the polytope is 1.

Then, representing each under-primer in the S&C model, for instance a_2, falls on one of these two cases:

- $a_2 = a_1$: in that case, the relation $f_{1/2}$ is the identity, so $C(a_2|\{a_1\}, P) = 0$.
- $a_2 \neq a_1$: it is necessary to represent the new element a_2 with $f_{1/2} \neq id$, so $C(a_2|\{a_1\}, P) = 1$.

[5]The primer is in fact the only element without antecedent as deleting an element implies the deletion of its successors.

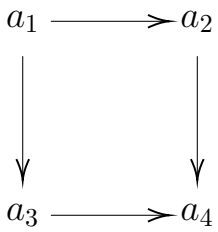


Figure 5: Square system.

The same principle applies for a_3 with relation $f_{1/3}$.
Finally, a_4 is evaluated in comparison with the fictive element $\hat{a_4} = f_{1/2}f_{1/3}.a_1$:

- If $\hat{a_4} = a_4$, the contrast is null. Hence, a_4 is encoded with the identity relation, yielding $C(a_4|\{a_1, a_2, a_3\}, P) = 0$.
- If $\hat{a_4} \neq a_4$, the contrast relation needs to be encoded to model a_4, and $C(a_4|\{a_1, a_2, a_3\}, P) = 1$.

Here, because $\hat{a_4} = f_{1/2}.a_3$, checking if the contrast is null is equivalent to checking if $f_{1/2}.a_3 == a_4$, or, differently written, if the relation between a_1 and a_2 is equal to the relation between a_3 and a_4, *i.e.* $f_{1/2} == f_{3/4}$.
In this test, a_4 is evaluated via its antecedent a_3, and by comparing the relation $f_{3/4}$ with the parallel arrow starting from the primer ($f_{1/2}$). We define as **pivot element** the extremity of this parallel arrow. In this case, the pivot of a_4 related to its antecedent a_3 is the element a_2, denoted p_4^3.

Definition 4.3 (Pivot)**.** In general, we define the pivot of an element a_i related to its antecedent a_j the extremity of a relation parallel to $f_{j/i}$ and having the primer as origin. It is denoted p_i^j.

By construction of polytopes, there always exists a pivot for elements which are not the primer or under-primers[6].

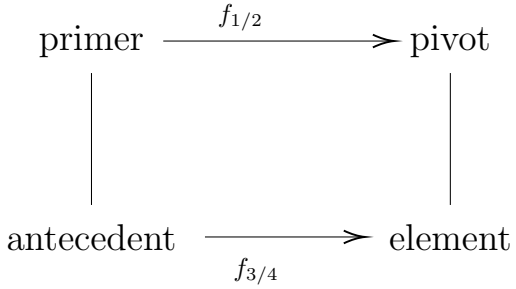


Figure 6: Square polytope with antecedent and pivot.

4.2.1 Equivalence of Both Couples Antecedent/Pivot for Square Systems

It is important to notice that, thanks to the commutativity of the relation group, in a square polytope, the choice of the antecedent for a_4 is not important. Indeed, let's compare both cases:

- Choosing a_2 as antecedent leads to choosing a_3 as pivot. Hence, testing the nullity of the contrast falls back to checking if $f_{1/3}.a_2 == a_4$. Yet, $a_2 = f_{1/2}.a_1$, so $f_{1/3}.a_2 = f_{1/3}f_{1/2}.a_1$, which leads to a test $f_{1/3}f_{1/2}.a_1 == a_4$.

[6]It is obvious in square polytopes, and it can be generalized to every regular polytope. It is also possible to generalize to non-deleted and non-added vertices in irregular polytopes, because, by design, an element is necessarily deleted if one of its antecedent is deleted.

- Choosing a_3 as antecedent leads to choosing a_2 as pivot. Hence, testing the nullity of the contrast falls back to checking if $f_{1/2}.a_3 == a_4$. Yet, $a_3 = f_{1/3}.a_1$, so $f_{1/2}.a_3 = f_{1/2}f_{1/3}.a_1$, which leads to a test $f_{1/2}f_{1/3}.a_1 == a_4$.

With commutativity of relations, $f_{1/3}f_{1/2}.a_1 = f_{1/2}f_{1/3}.a_1$, so both tests are equivalents.

Finally, when, for two different antecedents of an element, one is the pivot of the other, **it is equivalent to choose either one as the antecedent and the other as pivot**. This is the case for all square polytopes.

Algorithm 1 sums up the previous rules, as a complexity cost function for a 4-elements musical passage on a square polytope.

Algorithm 1 Compression cost of a square polytope

Input: 4 elements $\{a_1, a_2, a_3, a_4\} \in A^4$, and the abelian group G_r.
Output: Cost c

```
c = 1                                          ▷ Cost to encode a_1
for x = a_2, a_3 do
    if x == a_1 then
        c = c                                  ▷ Information is redundant.
    else
        c = c + 1                              ▷ This relation must be encoded.
if f_12 == f_34 then
    c = c                                      ▷ Information is redundant.
else
    c = c + 1                                  ▷ Contrast needs to be encoded.
return c
```

4.3 3-Polytope (Cube, 8 elements)

Now, let's consider 3-polytopes, *i.e.* cube polytopes, as presented in Figure 2. In this polytope, the primer is a_1, and the 3 sub-primers are a_2, a_3 and a_5.

Elements a_4, a_6 and a_7 have 2 antecedents shaping square polytopes, which is analogous to the previous case. However, the last element a_8 of this polytope leads to a new situation, as a_8 has 3 antecedents (a_4, a_6 and a_7) whose pivots are not antecedents (resp. a_5, a_3 and a_2). Here, each of the 3 antecedents defines a different square polytope with a different fictive element (a_1, antecedent, pivot, $\hat{a_8}$), as presented in Figure 7. Can these different S&C generate different fictive elements? And, if so, can a contrast be defined?

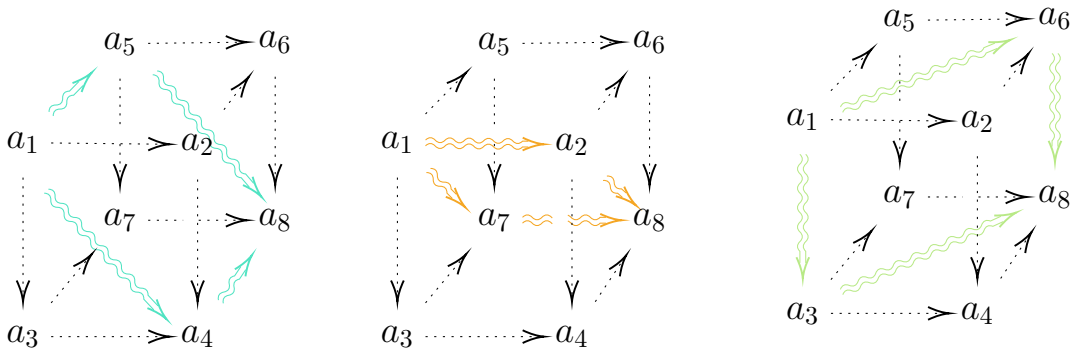


Figure 7: Different S&C generated by the 3 different antecedents of a_8.

Let us study these three cases. The "implication principle" for the contrast in the S&C means that, for a given antecedent, the relation antecedent/$\hat{a_8}$ is equal to the relation primer/pivot, or, equivalently thanks to commutativity, that the relation primer/antecedent is equal to the relation pivot/$\hat{a_8}$. Here, the three fictive elements are found as:

- $\hat{a_8^4} = f_{1/5}.a_4$ in the system $\{a_1, a_4, a_5, a_8\}$.
- $\hat{a_8^6} = f_{1/3}.a_6$ in the system $\{a_1, a_2, a_7, a_8\}$.
- $\hat{a_8^7} = f_{1/2}.a_7$ in the system $\{a_1, a_3, a_6, a_8\}$.

Applying the relation between the primer and the antecedent to the pivot is interesting, because, if the antecedent is itself contrasting in its own square system (for example, a_4 in the system $\{a_1, a_2, a_3, a_4\}$), this contrast is also assumed in the relation between the pivot and $\hat{a_8^4}$.

Hence, if several antecedents of a_8 are constrasting in their own square systems, different contrasts are assumed to construct the $\hat{a_8^i}$, leading to different fictive elements. There is here an ambiguity on the implication, which needs to be handled.

No antecedent is contrasting Firstly, let us consider the case where no antecedent of a_8 (a_4, a_6 and a_7) is contrasting. Here, these antecedents are equal to the composition of the relations primer/under-primer of their square systems ($a_4 = f_{1/2}f_{1/3}.a_1, a_6 = f_{1/2}f_{1/5}.a_1$ and $a_7 = f_{1/3}f_{1/5}.a_1$). Hence, the three fictive elements $\hat{a_8^i}$ are all equal to $f_{1/2}f_{1/3}f_{1/5}.a_1$, thanks to commutativity. Figure 7 can help the reader to understand this result.

Only one antecedent is contrasting Secondly, let us consider the case where **only one** antecedent is a contrast. In this case, two fictive elements are constructed without contrast as $f_{1/2}f_{1/3}f_{1/5}.a_1$ (as in the precedent case), and the third one replicates the contrast between the primer and this constratic antecedent, as $f_{1/2}f_{1/3}f_{1/5}\gamma.a_1$ (with γ denoting the contrastic relation).

Replicating this contrast holds more information than in the non-contrastic cases. Hence, as a rule, the fictive element constructed from the contrastic antecedent is considered as the only valid one.

In both previous cases, a unique valid fictive element $\hat{a_8}$ is constructed to evaluate a_8. The equality test $\hat{a_8} == a_8$ then determines the value of $C(a_8|\{a_1, ..., a_7\}, P)$.

More than one antecedent is contrasting Finally, when there are at least 2 antecedents with contrasts, it is unclear which fictive element should be chosen. In that case, a_8 is considered a constrast. Indeed, if a_8 does not admit a valid implication, it cannot be implied, so it is by nature a contrastive element. Hence, $C(a_8|\{a_1, ..., a_7\}, P) = 1$.

4.4 General Case, for Regular n-polytopes

The cube example introduces the case of an element with several antecedents, and the general case extends this principle.

To simplify visualizations, let us consider the 4-polytope case, represented in Figure 8. In this polytope, a_8 is no longer the only element with several couples antecedent/pivot, and is itself an antecedent of a_{16}.

When a_8 admits a unique valid fictive element $\hat{a_8}$ (only 0 or 1 contrast among its antecedents, as seen previously), the previous case can be extended by checking if a_8 is itself a contrast or not ($\hat{a_8} == a_8$).

When a_8 does not admit a fictive element, it is a contrast. In both cases, $\hat{a_{16}^8}$ can be constructed as $f_{1/8}.a_9$, and, counting how many constrastive antecedents a_{16} holds, the evaluation process for a_8, presented above, can be extended (with 4 antecedents instead of 3).

Hence, for any element a_i, if 0 or 1 of its antecedents is a contrast, a unique valid fictive element can be constructed, and $C(a_i|\{a_1, ..., a_{i-1}\}, P)$ depends on the relation $f_{\hat{i}/i}$. Otherwise, when several antecedents are contrasting, fictive elements are ambiguous, thus a_i is treated itself as a contrast, both for the cost $C(a_i|\{a_1, ..., a_{i-1}\}, P)$ and for its successors.

Definition 4.4 (Valid antecedents (set))**.** We call "valid antecedents" the set of all the antecedents of an element which can be used to construct a fictive element without ambiguity. Hence, this set can contain all antecedents of an element (if none of them is a contrast), only one (if it is the only contrast) or be empty. It is denoted V_i.

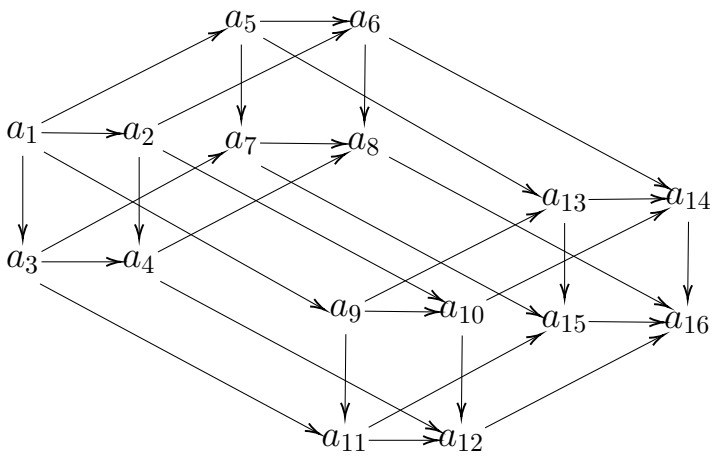


Figure 8: 4-polytope

The key point is to update the set of valid antecedents when facing a contrastic element: only this contrastic element has to be considered as valid for its successors. Concretely, this is made by intersecting each set of valid antecedents of its successors with this element (and its pivot). This indeed results in an empty set when several antecedents are contrastive.

The aforementioned process results in the general complexity cost function for regular polytopes, presented in Algorithm 2. In addition to previous definitions, let us denote S_i the set of successors of an element a_i.

Algorithm 2 Implication principle for an element a_i.

Input: a_i, element of a polytope, with a set of valid antecedents V_i, and relations $f \in G_r$, the current cost of the polytope c.
Output: Updated cost c

```
if ∃a_j ∈ V_i / f(a_1, a_j) = f(p_i^j, a_i) then
                ▷ Looking at its valid antecedents, and searching for an implication without contrast.
    c = c                                                    ▷ Information is redundant.
else                ▷ There is no implication without contrast (including the case where V_i is empty)
    c = c + 1                                                ▷ This element needs to be encoded.
    for s ∈ S_i do                                           ▷ Iterating over the successors of a_i
        V_s = V_s ∩ {a_i, p_s^i}
                ▷ Updating the antecedents of this successor, to keep only a_i and its pivot.
    return c
```

4.5 Irregular Polytopes

Finally, the complexity cost function can be extended to any irregular polytope. Starting with a n-polytope, an irregular polytope is constructed by deleting and/or adding another regular polytope of smaller dimension which contains the last element of the n-polytope.

As stated before, this condition ensures that, when deleting an element, all of its successors are also deleted. This also ensures that deletion does not break the previously designed rule. Thus, deletion only reduces the number of successors of some elements, but does not change the aforementioned rule.

Nonetheless, addition in a polytope adds a new case to the general rule, presented with help of Figure 9.

When an element is added to the polytope, the element on which it is attached is its antecedent (here for instance, a_4 is an antecedent of a_5, and equivalently for a_9 and a_{10}). However, these antecedence relations do not define a pivot element, as relations $f_{4/5}$ and $f_{9/10}$ do not have a parallel relation starting on the primer.

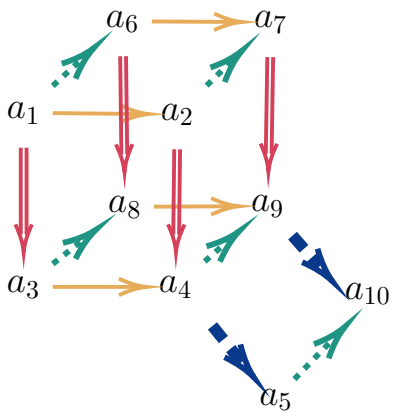


Figure 9: Visualization for addition

In that case, relations are compared with the identity function for the complexity cost. In practice, it can be obtained by considering that the pivot of a_5 related to its antecedent a_4 is the primer a_1, which follows the general rule.

Additionally, a_5 is also antecedent of a_{10}, and relation $f_{5/10}$ has a parallel relation $f_{1/6}$ starting from the primer, so a_6 is a pivot for a_{10} related to a_5, which follows the general rule.

4.6 Computing the Cost of a Polytope

Finally, Algorithm 3 presents the general complexity cost algorithm for a sequence of musical elements on a polytope.

Algorithm 3 Compression cost of a polytope.

Input: Polytope with vertices $\{a_i\} \in A, i = 1, ..., m$, and an abelian group G_r.
Output: Cost c

```
c = 1                                                    ▷ Cost to encode a_1
for i = 2, ..., m do
    V_i = {antecedents for a_i}   ▷ Initializing valid antecedents for all elements with all their antecedents
for i = 2, ..., m do                                     ▷ Iterating over the elements of the polytope
    if V_i = {a_1} then                                  ▷ If this element is an under-primer
        if a_1 = a_i then
            c = c                                        ▷ Information is redundant.
        else
            c = c + 1                                    ▷ This element needs to be encoded.
    else                                                 ▷ This element is not an under-primer
        if ∃a_j ∈ V_i / f(a_1, a_j) = f(p_i^j, a_i) then
                          ▷ Looking at its antecedents, and searching for an implication without contrast.
            c = c                                        ▷ Information is redundant.
        else   ▷ There is no implication without contrast (including the case where V_i is empty)
            c = c + 1                                    ▷ This element needs to be encoded.
            for s ∈ S_i do                               ▷ Iterating over the successors of a_i
                V_s = V_s ∩ {a_i, p_s^i}
                          ▷ Updating the antecedents of this successor, to keep only a_i and its pivot.
return c
```

5 Numerical Experiments

Algorithm 3 is developed in Python, along with a model handling polytopes, and is open-source[7] [4]. Results are based on the RWC Pop database [3].

5.1 Data

This algorithm has been tested in a same manner than C. Guichaoua in his PhD thesis [1], which introduces the polytopic analysis of music. Particularly, tests are conducted on the semiotic database of annotation for RWC Pop[8] [1, Chap.3.3]. This database contains beatwise aligned chord annotations, obtained from the initial annotations of the RWC Pop database [3] ("auto"), which were then manually corrected and homogenized by a human annotator ("manual").

In these annotations, each song is represented by a discretized sequence of perfect chords, synchronized on beats of the song. As a first attempt, silences were replaced with the previous chord (or the first chord of the song if silences are opening it). Defining a relation between a chord and a silence could be explored in future work.

5.2 Penalties

Section 4 presents the raw polytopic cost $C(S, P)$ for a musical passage S on a polytope P. Two penalty costs are added to this raw polytopic cost such that:

$$\mathcal{C}(S) = \min_{P} \left(C(S, P) + f_a(P)\right) + f_r(S) \tag{3}$$

iterating over all polytopes P containing $card(S)$ vertices, size of the musical passage.

5.2.1 Alteration Penalty f_a

A first penalty is applied to the polytope itself, related to its irregularities. In an information theory-like viewpoint, as presented in Section 4, a polytope can be defined by a quantity of information. Regular polytopes can be entirely described by their dimension, while altering a regular polytope (either by deletion or addition) requires to encode the shape and the position of the alteration.

In that sense, altering a polytope P increases the complexity. This increase is handled by adding a penalty $f_a(P) = p_a \in \mathbb{R}_+$ to the raw score when the polytope is irregular by either an addition or a deletion, and by adding $f_a(P) = 2p_a$ when the polytope is altered by both addition and deletion. Parameter p_a is fitted in experiments.

5.2.2 Regularity Penalty f_r

The second penalty considers the size of the segment. Indeed, as presented in [13], some segment sizes are more frequent than other in the RWC Pop database, particularly segments of 32 beats. Sargent et al. shows that adding a penalty prior in segmentation algorithms can enhance segmentation scores. For consistency with [1], we use the function $f_r(S) = p_r|card(S) - 32|$ as a regularization for the segment S of size $card(S)$. Parameter $p_r \in \mathbb{R}_+$ is fitted in experiments.

5.3 Scores

The goal of the task is to retrieve frontiers between structural segments, *i.e.* beats on which the segment is changing.

These estimated frontiers are compared with the annotation in order to compute True Positive, False Positive (wrong estimation of a frontier) and False Negative (frontier not found in estimation) rates. From this rates are computed Precision, Recall and F1-measure, as presented in [15]. A frontier is considered correct if it is exact or falls close enough (within a tolerance window) to an annotated frontier. These experiments were restricted to 0 and 3 beats tolerance windows, as in [1].

Technique		P_0	R_0	F_0	P_3	R_3	F_3	Computation time
MusicOnPolytopes [4]	$p_a = 0, p_r = 0$	50.3%	61.8%	55.1%	55%	68%	60.4%	3 $\frac{1}{2}$ hours
	$p_a = 3, p_r = 0.1$	68.2%	73.6%	70.6%	68.9%	74.5%	71.4%	3 $\frac{1}{2}$ hours
Results from [1]	$p_a = 0, p_r = 0$	-	-	43.7%	-	-	-	Not mentionned
	Optimal conditions*	-	-	69%	-	-	70%	Not mentionned
Code of [1], on author's laptop	$p_a = 0, p_r = 0$	35.8%	56.7%	43.3%	39.6%	62.9%	47.9%	8 hours
	Optimal conditions*	59.2%	63.4%	61.1%	61.2%	65.6%	63.2%	8 hours

Table 1: Numerical experiences on database "Manual". *Optimal conditions refer to the optimal conditions of [1], which are slightly different than in our model. C. Guichaoua indeed considered that p_a should be different when considering addition and deletion, leading to two parameters p_a^+ and p_a^-, and also that p_r should distinguish sizes larger and lower than 32, leading to two parameters p_r^+ and p_r^-. These optimal conditions hence refer to $p_a^+ = 2.25, p_a^- = 3, p_r^+ = 0, p_r^- = 0.125$.

Technique		P_0	R_0	F_0	P_3	R_3	F_3
MusicOnPolytopes [4]	$p_a = 0, p_r = 0$	29%	39.5%	33.1%	42.6%	59.8%	49.2%
	$p_a = 4, p_r = 0.2$	44.5%	47.1%	45.6%	56.2%	60%	57.8%
Results from [1]	$p_a = 0, p_r = 0$	-	-	-	-	-	-
	Optimal conditions*	-	-	37.4%	-	-	55.2%
Code of [1], on author's laptop	$p_a = 0, p_r = 0$	23.6%	39%	28.9%	35%	59.5%	43.4%
	Optimal conditions*	41.7%	44.4%	42.8%	53.9%	57.7%	55.5%

Table 2: Numerical experiments on the "Auto" database. *Optimal conditions refer to the optimal conditions of [1], equal to: $p_a^+ = 2.5, p_a^- = 2.5, p_r^+ = 0, p_r^- = 0.125$.

Database	Technique		P_0	R_0	F_0	P_3	R_3	F_3
Manual	Triad circle	$p_a = 0, p_r = 0$	50.3%	61.8%	55.1%	55%	68%	60.4%
		$p_a = 3, p_r = 0.1$	68.2%	73.6%	70.6%	68.9%	74.5%	71.4%
	Tonnetz	$p_a = 0, p_r = 0$	50%	61.2%	54.7%	55.3%	68.4%	60.7%
		$p_a = 3.5, p_r = 0.1$	67.1%	72.5%	69.5%	68%	73.6%	70.5%
Auto	Triad circle	$p_a = 0, p_r = 0$	29%	39.5%	33.1%	42.6%	59.8%	49.2%
		$p_a = 4, p_r = 0.2$	44.5%	47.1%	45.6%	56.2%	60%	57.8%
	Tonnetz	$p_a = 0, p_r = 0$	27.1%	36.6%	30.8%	41.1%	57.7%	47.5%
		$p_a = 4, p_r = 0.3$	44.8%	46.3%	45.4%	56.5%	58.7%	57.4%

Table 3: Results of MusicOnPolytopes [4]. Comparison between triad circle and tonnetz relations.

Results presented in tables 1 and 2 are computed using the triad circle model of relations, which is common to both works. Results obtained with the new *MusicOnPolytopes* toolbox are higher than those of [1]. At this time, the differences are difficult to explain.

In addition, when running the code of [1] (obtained from C. Guichaoua himself), the results on the Manual database are worst than the ones presented in [1]. These results may be due to downgrading or modifications of external libraries since its initial development, but are also hard to explain.

In addition, Table 3 compares segmentation results obtained either with the triad circle or the tonnetz as relation model. Results are not significantly different between the two models of relations, but the tonnetz obtains generally worst results than the triad circle.

6 Conclusion

In conclusion, this article presents a new code framework, in Python, to compute polytopic analysis of music, introduced in [1]. This framework shows interesting results when applied on the structural segmentation task of symbolic music.

This work shows an improvement in performance compared to those obtained by C. Guichaoua in [1], and calls for further development. In particular, an exciting lead would be the development of a new group of relation G_r for discretized audio signals, in order to extend this work for the structural segmentation of audio signals.

References

[1] C. Guichaoua, *Modèles de compression et critères de complexité pour la description et l'inférence de structure musicale*. PhD thesis, 2017.

[2] F. Bimbot, E. Deruty, G. Sargent, and E. Vincent, "System & contrast: A polymorphous model of the inner organization of structural segments within music pieces," *Music Perception: An Interdisciplinary Journal*, vol. 33, no. 5, pp. 631–661, 2016.

[3] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, Classical and Jazz Music Databases," in *ISMIR*, vol. 2, pp. 287–288, 2002.

[4] A. Marmoret, J. E. Cohen, and F. Bimbot, "MusicOnPolytopes," Feb. 2021.

[5] L. B. Meyer, *Emotion and meaning in music*. University of chicago Press, 1956.

[6] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

[7] E. Narmour, *The analysis and cognition of basic melodic structures: The implication-realization model*. University of Chicago Press, 1990.

[8] M. T. Pearce, D. Müllensiefen, and G. A. Wiggins, "Melodic grouping in music information retrieval: New methods and applications," in *Advances in music information retrieval*, pp. 364–388, Springer, 2010.

[9] D. Meredith, "Music analysis and kolmogorov complexity," in *XIX Colloquio di Informatica Musicale*, 2012.

[10] S. Lattner, *Modeling Musical Structure with Artificial Neural Networks*. PhD thesis, Johannes Kepler University Linz, 2019.

[11] C. Louboutin, *Multi-scale and multi-dimensional modelling of music structure using polytopic graphs*. PhD thesis, Université Rennes 1, 2019.

[7] https://gitlab.inria.fr/amarmore/musiconpolytopes

[8] which can be found at https://gitlab.inria.fr/amarmore/rwc_quaero_annotations

[12] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-based music structure analysis: Current trends, open challenges, and applications," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[13] G. Sargent, F. Bimbot, and E. Vincent, "Estimating the structural segmentation of popular music pieces under regularity constraints," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 2, pp. 344–358, 2016.

[14] R. Cohn, "Neo-riemannian operations, parsimonious trichords, and their" tonnetz" representations," *Journal of Music Theory*, vol. 41, no. 1, pp. 1–66, 1997.

[15] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 318–326, 2008.

A Computational Representation of Polytopes

A polytope can be represented either by its geometrical model (n-hypercube with alteration), or by a symbolic representation, more suited for computational treatment.

In his thesis [1], C. Guichaoua uses binary trees to model polytopes, where leafs represent final elements.

In MusicOnPolytopes, polytopes are represented by nested lists. An element of a polytope is represented by a "1", and every dimension represents a level of nesting of this element. For instance, a 1-polytope, linking two elements, is represented as [1,1], and a 2-polytope, with 4 elements, is represented as [[1,1], [1,1]].

For irregularities, a deletion of an element is represented by the deletion of a "1" in this list, for instance the 2-polytope with the 0-polytope deletion is represented as [[1,1], [1]]. An addition is represented by a tuple, signifying on which vertex is attached the new element, for instance the 1-polytope with the 0-polytope addition is represented as [1, (1,1)].

These general polytopes can then be adapted to a particular musical sequence (for instance [[Ab, Ab],[Gm,Gm]]), or extended to indexed polytope, where each element represents the index of the element in the polytope (for instance [[0,1], [2,3]]).

A tutorial notebook is present with the code[9].

[9] https://gitlab.inria.fr/amarmore/musiconpolytopes/-/blob/master/Notebooks/Tutorial - Handling polytopes.ipynb

UNIVERSITÉ DE RENNES 1

Titre : Paradigmes d'Apprentissage Automatique Non-Supervisés pour les Représentations de la Similarité et de la Structure Musicale.

Mot clés : Segmentation Structurelle, Musique, Apprentissage Automatique Non-Supervisé et Méthodes d'Optimisation

Résumé : La structure musicale, définie comme la représentation simplifiée de l'organisation d'un morceau de musique, est un concept musicologique important mais néanmoins complexe à estimer automatiquement. Cette thèse présente de nouvelles méthodes pour estimer automatiquement la structure musicale, se focalisant sur l'étude à l'échelle de la mesure musicale. Par le développement d'un nouvel algorithme de segmentation (appelé "CBM") et par l'étude et la comparaison de différentes méthodes de compression non supervisées (allant de l'algèbre linéaire et multilinéaire aux réseaux de neurones), les paradigmes introduits dans cette thèse permettent d'obtenir des résultats quantitatifs dépassant l'Etat-de-l'Art non supervisé actuel et se rapprochant de l'Etat-de-l'Art global, issu de méthodes d'apprentissage avec supervision. En particulier, les méthodes décrites dans cette thèse étant non supervisées, l'estimation ne repose pas sur des bases de données annotées, permettant ainsi de mitiger les biais liés à l'ambiguïté et à la subjectivité (inhérents à la structure musicale), tout en limitant le perte en performance par rapport aux meilleures méthodes supervisées. Enfin, certaines méthodes étudiées dans cette thèse (en particulier la décomposition nonnégative en Tucker) permettent d'extraire automatiquement des parties interprétables de la chanson qui pourraient être utilisées pour d'autres tâches que l'estimation de structure, et s'intégrer dans le développement d'algorithmes interprétables d'apprentissage automatique profond, sujet de recherche majeur aujourd'hui.

Title: Unsupervised Machine Learning Paradigms for the Representation of Music Similarity and Structure.

Keywords: Structural Segmentation, Music, Unsupervised Machine Learning and Optimization methods

Abstract: Musical structure, defined as a simplified representation of the organization of a song, is an important musicological concept, but hard to automatically estimate. This thesis presents new methods to automatically estimate the structural segmentation of a song, focusing the study of music at the barscale. By developing a new segmentation algorithm (called "CBM") and by comparing several unsupervised compression schemes (from linear and multilinear algebra to neural networks), paradigms introduced in this thesis result in segmentation performance outperforming those of the unsupervised State-of-the-Art methods and almost similar with those of the global State-of-the-Art, obtained with supervised machine learning algorithms. In particular, as the methods described in this thesis are unsupervised, the estimation do not rely on annotated data, lowering the bias in the estimates related to ambiguity and subjectivity (inherent to musical structure) while limiting the loss in performance compared to the best supervised methods. In addition, some of the methods studied in this thesis (in particular Nonnegative Tucker Decomposition) allow to extract automatically interpretable parts of a song which may be used for other task than the estimation of structure, and participate in the development of interpretable machine and deep learning algorithms, which is a major field of research nowadays.