Representation learning for symbolic music

Mathieu Prang

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité Informatique
ED130 - Ecole doctorale Informatique, Télécommunications et Electronique (Paris)
Sciences et Technologie de la Musique et du Son (UMR 9912)
Institut de Recherche et de Coordination Accoustique Musique
Equipe Représentations musicales.

REPRESENTATION LEARNING FOR SYMBOLIC MUSIC

MATHIEU PRANG

SUPERVISÉ PAR : PHILIPPE ESLING
DIRIGÉ PAR : CARLOS AGON

Defended on June 7th, 2021

JURY :

Frédéric Bimbot (Reviewer)
Anna Jordanous (Reviewer)
Jean-Pierre Briot
Simon Colton
Florence Levé
Geoffroy Peeters
Carlos Agon (Director)
Philippe Esling (Supervisor)

Music is enough for a lifetime, but a lifetime is not enough for music.

— Sergei Rachmaninov

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.

— Marie Curie

ABSTRACT

A key part in the recent success of deep language processing models lies in the ability to learn efficient word embeddings. These methods provide structured spaces of reduced dimensionality with interesting metric relationship properties. These, in turn, can be used as efficient input representations for handling more complex tasks. In this thesis, we focus on the task of learning embedding spaces for polyphonic music in the symbolic domain. To do so, we explore two different approaches. The first one is inspired by the work done in the Natural Language Processing (NLP) field and relies on prediction tasks, while the second is based on the latent space of Variational Auto-Encoders (VAE).

We introduce an embedding model based on a convolutional network with a novel type of self-modulated hierarchical attention, which is computed at each layer to obtain a hierarchical vision of musical information. We show that this model provides a strong increase in prediction accuracy on all reference symbolic music prediction datasets. We further evaluate the quality of the resulting embedding spaces by analyzing metric relationships between musical elements. We show that interesting geometrical structures are naturally discovered by the model, which reflect music theory properties. However, we notice that similar elements might be encoded in widely different regions of the embedding revealing a lack of control over the latent space topology during training.

In order to overcome this effect, we propose another system based on VAEs, a type of auto-encoder that constrains the data distribution of the latent space to be close to a prior distribution. As polyphonic music information is very complex, the design of input representation is a crucial process. Hence, we introduce a novel representation of symbolic music data, which transforms a polyphonic score into a continuous signal. To do so, we map MIDI pitches to prime frequencies with a random imaginary part, allowing to perform an inverse Fourier transform with minimal resolution. We evaluate the ability to learn meaningful features from this representation from a musical point of view and conduct an extensive benchmark against recent polyphonic symbolic representations. We show that our signal-like representation improves the stability of learning, leading to better reconstruction and disentangled features. This improvement is reflected in the metric properties of the space learned from our signal-like representation, which better correlates

with music theory properties.

Finally, we show the potential of the resulting embedding spaces through the development of several creative applications used to enhance musical knowledge and expression, through tasks such as melodies modification or composer identification.

RÉSUMÉ

Un élément clé du récent succès des modèles d'apprentissage profond de traitement du langage réside dans la capacité à apprendre des "embeddings" de mots efficaces. Ces méthodes fournissent des espaces vectoriels structurés de dimension réduite ayant des relations métriques intéressantes. Ceux-ci, à leur tour, peuvent être utilisés comme des représentations d'entrée efficaces pour traiter des tâches plus complexes. Dans cette thèse, nous nous concentrons sur la tâche d'apprentissage d'espaces "d'embedding" pour la musique polyphonique dans le domaine symbolique. Pour ce faire, nous explorons deux approches différentes. La première s'inspire des travaux réalisés dans le domaine du traitement du langage et repose sur des tâches de prédiction, tandis que la seconde est basée sur l'espace latent des encodeurs automatiques variationnels (VAE).

Dans ce manuscrit, nous introduisons un modèle d'embedding basé sur un réseau convolutif avec un nouveau type de mécanisme d'attention hiérarchique automodulée, qui est calculée à chaque couche afin d'obtenir une vision hiérarchique de l'information musicale. Nous montrons que ce modèle permet une augmentation importante de la précision de prédiction sur tous les jeux de données de musique symbolique de référence. Nous évaluons ensuite la qualité des espaces d'embedding en analysant les relations métriques entre les éléments musicaux. Nous montrons que des structures géométriques qui reflètent les propriétés de la théorie musicale sont découvertes naturellement par le modèle. Cependant, nous remarquons que des éléments similaires peuvent être encodés dans des régions très différentes de l'espace, ce qui révèle un manque de contrôle des propriétés de l'espace latent pendant l'apprentissage.

Afin de pallier à cet effet, nous proposons un autre système basé sur les VAE, un type d'auto-encodeur qui contraint la distribution des données de l'espace latent à être proche d'une distribution préalablement choisie. La musique polyphonique étant un type d'information complexe, le choix de la représentation d'entrée est un processus crucial. Nous introduisons donc une nouvelle représentation de données musicales symboliques, qui transforme une partition polyphonique en un signal continu. Pour ce faire, nous remplaçons les hauteurs MIDI par des nombres premiers avec une partie imaginaire aléatoire permettant d'effectuer une transformation de Fourier inverse de petite résolution. Nous évaluons la capacité d'apprendre des caractéristiques musicales intéressantes à partir de cette repré-

sentation et nous effectuons une comparaison approfondie avec les principales représentations de la musique polyphonique proposées dans la littérature. Nous montrons que notre représentation "signal-like" améliore la stabilité de l'apprentissage, ce qui conduit à une meilleure reconstruction et à des caractéristiques mieux séparées. Cette amélioration se reflète dans les propriétés métriques de l'espace qui présente une meilleure corrélation avec les propriétés de la théorie musicale.

Enfin, nous montrons le potentiel de nos espaces d'embedding à travers le développement de plusieurs applications créatives utilisées pour améliorer la connaissance et l'expression musicales, à travers des tâches telles que la modification de mélodies ou l'identification de compositeurs.

ACKNOWLEDGMENTS

First and foremost I want to thank my supervisor Philippe Esling for showing me its confidence by awarding me this research project. His friendly guidance and expert advice have made this work profoundly pleasant and captivating. Besides the enthusiastic vision he has for a wide variety of things, I could benefit from his vast scientific knowledge that allowed me to improve my competences in many fields. I am also very grateful for his unfailing humanity that drives him to care for those around him.

I am indebted to Carlos Agon who directed this thesis with reliability and positivism, ensuring that our exchanges were always pleasant. My sincere thanks also goes to all future and already seasoned researchers at IRCAM who helped me through their insightful comments. Especially the members of the ACIDS team, Léopold Crestel, Axel Chemla-Romeu-Santos, Tristan Carsault, Adrien Bitton, Jean-François Ducher, Constance Douwes, Théis Bazin, Ninon Devis, Antoine Caillon, Cyran Aouameur and Clement Tabary, who showed genuine interest in my work and contributed largely to it with many brilliant ideas.

I take this opportunity to express my sincere gratitude to all my friends with whom I spend precious time that gives me the energy I need to go ever further. I have a special thought for Hadrien Foroughmand who provided me unfailing support and so many unforgettable moments from our early years of higher education to the end of our respective theses.

Last but not the least, a special thanks to my family for their love and incredible support. For my parents who raised me with a love of music and sciences and encouraged me in all my pursuits. For my mother whose strength of mind continues to impress me over the years and whose love and dedication to others constitute my foremost inspirational force. For my two outstanding brothers whose wise guidance has given so much to me over the years.

And most of all, for my loving, supportive, and awesome Camille who give me the strength to overcome difficulties and keep the faith on me.

Thank you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

NLP Natural Language Processing

NN Neural Newtork

RNN Recurrent Neural Network

LSTM Long-Short Term Memory

CNN Convolutional Neural Network

VAE Variational Auto-Encoder

ML Machine Learning

AM Attention Mechanism

MIDI Musical Instrument Digital Interface

LISP Locator/Identifier Separation Protocol

XML eXtensible Markup Language

GD Gradient Descent

SGD Stochastic Gradient Descent

MSGD Mini-Batch Gradient Descent

ReLU Rectified Linear Unit

MSE Mean Squared Error

AE Auto-Encoder

PCA Principal Component Analysis

t-SNE t-distributed Stochastic Neighbor Embedding

NADE Neural Autoregressive Distribution Estimator

RBM Restricted Boltzmann Machine

HAM Hierarchical Attention Modulation

BCE Binary Cross Entropy

MAESTRO MIDI and Audio Edited for Synchronous TRacks and Organization

CBOW Continuous Bag-Of-Word

GloVe Global Vectors for word representation

1 INTRODUCTION

When hearing a musical piece or reading its score, human beings are able to implicitly interpret sets of intricate information inside this input data. Indeed, thanks to musical experiences developed throughout our lives, we can easily link between unknown audio signals and known concepts such as musical genre, tempo or harmony. These concepts can be thought of as high-level abstractions, oppositely to low-level ones like acoustic signal values.

Over the past decades, the field of computer music has precisely addressed problems surrounding the analysis of musical concepts. Indeed, it is only by first understanding this type of information that we can provide more advanced tools of analysis and composition and methods that improve our musical knowledge. Nowadays, a wide variety of approaches have been pursued, with a notable recent rise of inspirations from the machine learning field.

Throughout this thesis, we have focused on a very promising idea proposed in machine learning, the idea of *embedding spaces*. This particular method consists of designing models, which are able to *project* musical data into a smaller space, where the metric properties reflect high-level musical theory concepts. These spaces are very powerful as they can be used as analytical and even compositional tools by themselves, but also as an input representation for complex tasks. Indeed, as these spaces provide an efficiently-organized musical information, they can allow to tackle more intricate issues in an easier way.

In this chapter, we define the context and goals of this thesis. We will first explain our motivations inside the specific field of computer music, by highlighting major breakthroughs that have permitted our research, but we also underline the remaining challenges to tackle. Finally, we outline the global structure of this manuscript and introduce our main contributions.

1.1 MOTIVATIONS

Concomitantly with the appearance of computers, a large interest in their use for musical purposes has emerged. Their computational aspects appeared as a way to free ourselves from physical boundaries, creating novel musical instru-

ments able to produce large arrays of sound with uncommon *timbral* properties. This curiosity quickly spread to the creation of tools for composing, recording and analyzing music. For more than half a century, artists and scientists have been working together on research issues and artistic works, driving this field to move hand-in-hand with the expansion of computational resources. These technical possibilities have been seized by artists, motivating their need for new tools from researchers and engineers. Hence, computer music has played a major role in reshaping modern music, even becoming one of the main pillar in popular music.

Although a wide variety of scientific challenges have gradually emerged in this field, one of the most crucial remains in adequate representations of music that can be processed efficiently by computers. Indeed, musical notation has evolved throughout the ages until becoming a complete language in its own right. However, due to their inner structures and construction, computers are not well fit to directly process this multi-dimensional type of data with complex shapes and unique symbols. Therefore, it is compulsory to first convert symbolic music into a machine-readable format. For this reason, the representation of music must be transformed and thought-of differently to be effective in our domain.

This specific question has stirred up a huge interest in the research community, leading to a wide variety of representations, that we refer here as *musical spaces*. Despite their algebraic nature tailored for efficient processing, their real value can be assessed through the underlying organization of musical theory information. Although different types of efficiency are usually targeted, all these formalization share a similar process of development, trying to build spaces through known mathematical rules. Unfortunately, this usually constrains these spaces to a restricted scope of application in Western music, and are often not easily applicable to other contexts.

However, novel possibilities have emerged through the rise of the recent *deep learning* field. Although the idea of mimicking the behavior of the human neuron through computers dates back to the middle of the 20^{th} century, the lack of computational capacities and sufficient amount of data has greatly impeded the usability of this framework. However, over the last decade, technological advances have allowed the training of so-called *deep* architectures, where artificial neurons are stacked in successive layers. These achievements have brought about a revolution in computer science, opening up entire fields of research in various domains. Out of all these new challenges, one question has aroused particular interest: *could a computer be able to learn a meaningful representation of music by itself?*

Recently, different breakthroughs in the Natural Language Processing (NLP) field have provided the building blocks towards our overarching goal. Particularly relevant to our work is the huge step forward in the development of *word embedding spaces*. In this context, large datasets of sentences are used to understand the relationships between words. The goal is to find a space where words are represented as points (vectors), whose distance relationships mirror their semantic similarity. By using these vectors as a representational basis for other machine learning tasks, scientists made colossal improvements and opened up possibilities for a wide variety of powerful applications. For instance, Palangi et al., 2016 used word embedding to perform efficient document retrieval or web search. Similarly, Tang et al., 2014 developed a tool that classify the messages from Twitter according to their sentiments.

In our context, structural similarities between the language and music field could hypothetically share some logical equivalence. Indeed, a sentence is composed by words hierarchically located, akin to a melody, which is composed by notes. Moreover, this kind of learned space could also be very valuable for the musical analysis and composition field, as a potential analysis and knowledge inference tools, but also as a new representation for other creative application. Moreover, this continuous space could provide direct ways for performing melody generation or transformation. One of the most interesting challenge would be to link these spaces with perception or signal processing knowledge as it has been done in other fields (Aytar, Vondrick, and Torralba, 2016; Karpathy and Fei-Fei, 2015; Kiros, Salakhutdinov, and Zemel, 2014; Mroueh, Marcheret, and Goel, 2015). Armed with these combined spaces, we could find some novel relevant features about music and develop powerful classification or recommendation tools. Setting out from these intriguing premises, we decided to devote our work on these embedding spaces for symbolic music.

1.2 DISSERTATION ORGANIZATION AND MAIN CONTRIBUTIONS

First, Chapter 2 is dedicated to the presentation of the state-of-the-art related to our work. In Section 2.2, we present the fundamentals of musical notation, while highlighting its importance in the development of music throughout different eras. Then, we explain how notation has been adapted to computer science with efficient digital representations and the powerful concept of *musical spaces*. In Section 2.3, we detail the core principles of the Machine Learning (ML) field. We start by exposing basic concepts, allowing to develop the mechanisms of the specific models that have been used during this thesis. Finally, we provide in Section 2.4 a global

overview of the embedding spaces framework from its foundation to the most recent proposals based on VAE in Section 2.5.

Chapter 3 presents our first contribution on learning musical embedding spaces. This method derived from NLP relies on the prediction of incoming musical events in a given context. First, we propose in Section 3.2 our model architecture based on CNN and Long-Short Term Memory (LSTM) networks. We also introduce a new Attention Mechanism (AM), which greatly improves the overall performances of our system allowing our model to obtain better results than the state-of-the-art in musical events prediction. In addition to the prediction results, we also analyze in Section 3.3 the visualizations of projected data in our learned embedding space. We see that, despite some structural inconsistencies, several musical relations are captured, which holds great promise for further developments of our research.

In Chapter 4, we focus on our second method, which relies on the VAE framework. As shown in the literature, these kinds of approaches are highly dependent of the input representation. Thus, we introduce in Section 4.3 a novel approach for representing polyphonic excerpts of music called the *signal-like representation*. We demonstrate its efficiency by conducting an extensive benchmark against recent polyphonic symbolic representations. We show that our signal-like representation improves the stability of learning, leading to better reconstruction and disentangled features. This improvement is reflected in the metric properties of the space learned from our signal-like representation, which correlates with music theory properties, as shown in Section 4.4.

Chapter 5 presents several applications of the embedding space, which are made possible thanks to their structured natures. In Section 5.2, we propose a composers classification tool trained with our learned representation as input, which can be used for dataset labelling or unknown score analysis. We show in Section 5.3.1 how our spaces can be used to compute *attribute vectors*, which can be used to make targeted changes in musical excerpts. Lastly, we present in Section 5.3 a tool which aims to enhance the creativity of composers by helping them to find interesting changes in their melodies.

Finally, Chapter 6 summarizes our work and contributions, by discussing our results and opening directions for possible future works.

2

OVERVIEW

2.1 INTRODUCTION

In this chapter, we provide an overview on all the fundamentals concepts useful for further reading of the manuscript. We also present the state of the art methods related to the learning of embedding spaces for symbolic music.

In Section 2.2, we first highlight the importance of the representation in music and how it has become a major challenge in the computer science field. We present the three main generic and extendable representations for polyphonic music, namely the *piano-roll*, *MIDI-like* and *NoteTuple* representation. Finally, we introduce the concept of rule-based *musical space* where items are positioned according to music theory rules allowing the inference of musical knowledge, new composition processes and an easier manipulation of scores.

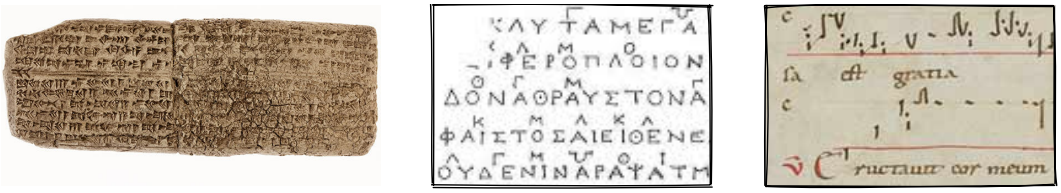
The following Section 2.3 is dedicated to the ML framework. To begin with, we develop the basics of this field including its formal description and the training process with its ensuing mechanics. Then, we are going further by introducing all the advanced models that we have used during this thesis.

We explain the apparition and formalism of *embedding spaces* in Section 2.4. We demonstrate the merits of this approach through the presentation of the two most outstanding models of the NLP field. Finally, in Section 2.5, we show the work done to apply it on musical data which can be divided in two categories. The first one, referred as *prediction-based* consists of mimicking the word embeddings mechanics which rely on events prediction while the second one, the *VAE-based*, is built on Variational Auto-encoding.

2.2 SYMBOLIC MUSIC REPRESENTATION

2.2.1 *Music as symbol*

Music can be described as a set of sounds organized in time, that most of the civilizations have tried to transcribe into written format since very ancient times. Therefore, it is difficult to precisely date the first appearance of musical symbols



(a) 1400 BC (b) 128 BC (c) 11th century

Figure 2.1: Evolution of the musical notation across the ages. (a) The first score engraved in cuneiform writing on a clay tablet which date back to to 1400 BC. (b) Ancient Greece musical notation written in 128 BC where the letters represent the notes (transcribed by Annie Belis). (c) The Neumes are symbols which encode simple melodies along a line representing a fixed sound (11th century).

but it seems that the first *score* discovered traces back to 1400 BC. This music is engraved in cuneiform writing on clay tablets (see Figure 2.1). Even though the interpretation of this notation system is still under debate, it is clear that it provides instructions for performing music. We also learn from its reading that the music was composed in harmonies of thirds and using a diatonic scale (Bosseur, 2005).

In Ancient Greece, the foundations of one of the first notation specifically tailored for transcribing music has been laid by the music theorist Alypius. He created two alphabets – one is dedicated to vocals and the other to instruments – where the letters represent the notes and are distorted to emphasize musical variations (see Figure 2.1). The use of such a notation already required a significant amount of knowledge that very few people had access to, leading to the creation of a very simplified system for the daily practice of music composed only of syllables. Here, we can assume that this has ushered in the emergence of a schism between the so-called "art music" and the popular music.

Although taken up and completed by the Romans and then by the Byzantines, we had to wait until the beginning of the Middle Ages to see a significant breakthrough in musical notation with the appearance of *the neumes*. These symbols take the form of musical figures applied on syllables that do not encode individual notes but simple melodies. At first they do not indicate precise intervals between notes, only grave or acute accents allow to differentiate the pitch. Then in the course of the 10th century, the idea of drawing a line representing a fixed sound, above and below which the neumes were ordered, make its appearance (see Figure 2.1). Fifty years later, a red line represented the F, a yellow one the Ut, these were the first musical stave lines.

The Middle Ages were marked by two notable periods corresponding to distinct compositional styles. The first one, called "*ars antiqua*", has witnessed the rise of the polyphony where several instruments play simultaneously. At this point, it became a necessity to accurately transcribe pitches for the global harmony between voices and rhythm for the coordination between instrumentalists. Therefore, the number of stave lines has increased to five, as it is today, except for the Gregorian chants where the smaller pitch range can be handled with only four lines. Single notes are now represented by little black squares arranged on the staff, with or without tail according to their duration.

Until then, the ternary rhythms (which divide time by three, creating a revolving, waltzing pulse) are predominant since they refer to the *Holy Trinity* [1]. However, during the "*ars nova*", the second major artistical trend of the Middle Ages, the rhythm begins to be theorized thus triggering a further improvement in the notation system (Apel, 1961). Indeed, the written note now takes different shapes – square, rectangle, diamond – based on its duration. These enhancements have democratized the binary rhythms which divide the time by two, giving a steady and regular character to the music.

However, we have to wait until the 15th century for a real standardization of the music writing with the invention of the printing process. The square notes will give way to round notes that are more suitable for engraving with a chisel. We also see the emergence of the bar lines that give rhythm a central role and reinforce the mathematical aspect of the music.

In the centuries that followed, Western musical notation became more complex and spread throughout the world which made possible to fix in writing traditional musics hitherto transmitted orally. However, this had the consequence of denaturing some of them, as in China for example, music written with European rules sounded much more like Western music than traditional Chinese music.

We can see that the representation of music as symbols itself has been a central question in the history of music. In that sense, musical notation could be thought of as a model which enables us to reason and think about music.

1 Holy trinity, in Christian doctrine, the unity of Father, Son, and Holy Spirit as three persons in one Godhead.

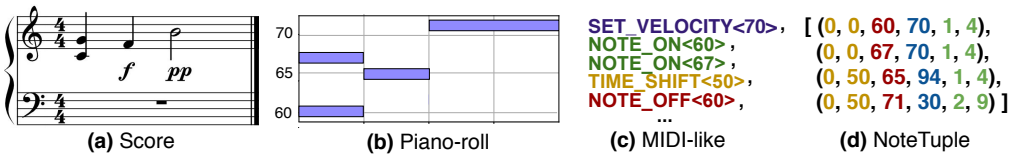


Figure 2.2: Representations for symbolic music learning from scores (a). The *piano-roll* (b) is the most widespread representation. The *MIDI-like* (Oore et al., 2018) representation (c) encodes it as a sequence of events, while the *NoteTuple* (Hawthorne et al., 2018) representation (d) encodes the time offset, MIDI pitch number, velocity and two values for the duration.

2.2.2 *Symbolic music representations for computer science*

Nowadays, with the apparition of the digital era, multiple machine-readable scores formats have been developed, the most prominent being the Musical Instrument Digital Interface (MIDI). This type of digital data format allows to treat music through its symbolic representation since it is based on a finite alphabet of symbols. A MIDI file encodes information for the different notes, durations and intensities through numerical messages with a pre-defined temporal quantification (a subdivision of a quarter note). Hence, this format has been widely used in computer music research as it allows a compact representation of music. Other formats have been developed using for instance Locator/Identifier Separation Protocol (LISP) (Assayag et al., 1999) or eXtensible Markup Language (XML) (Good et al., 2001).

In order to fully benefit from the growing computational capability of the modern machines, these digital scores have to be encoded into more algebraic structures like vectors and matrices. Some representations are specific to a precise task, like the *couple* representation (Hadjeres, Pachet, and Nielsen, 2017) used for representing four-voices chorales written by Jean-Sebastian Bach. Despite the tremendous generation results obtained from it, we focus in this chapter on more generic proposals which may fit well for any kind of musical data. We present here the three main representations of the literature.

2.2.2.1 *Piano-roll*

The most common way to represent polyphonic music is through the *piano-roll* representation. Here, time is discretized with a reference quantum (typically a fraction of the quarter note) to provide a matrix $P(n, t)$ that represents note activation in musical sequences. An example is depicted Figure 2.2. This representation fits for every kind of music as it properly handles polyphony. However, the resulting matrices are relatively high dimensional (88 to 128 dimensions per time step, per voice) and highly repetitive due to its discrete nature. Moreover, because of the

typically small amount of notes played simultaneously, these matrices are usually highly sparse.
For these reasons, this representation raises several issues for learning, which warranted the definition of alternate approaches.

2.2.2.2 *MIDI-like*

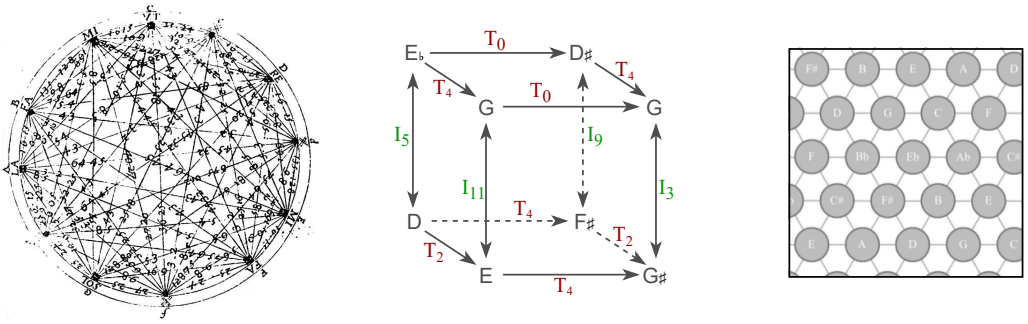
The first alternative representation has been proposed in Oore et al., 2018. This *MIDI-like* approach relies on an event-based vocabulary composed by four main MIDI events. The `NOTE_ON` event (composed of 128 possible sub-events) indicates the start of the corresponding MIDI note. Similarly, the `NOTE_OFF` event signifies the end of a played note. The `TIME_SHIFT` event is composed by 125 sub-events and moves the time step forward by increments of 8 ms up to 1 second. Finally, the `SET_VELOCITY` event counts 32 sub-events which changes the velocity applied to all subsequent notes until the next velocity event. Hence, the resulting representation of an input piece is a variable-length sequence of discrete events taken from this vocabulary. We can see an example Figure 2.2 This representation can handle any form of music with polyphony and variable number of voices or time signatures. However, as the MIDI-like representation relies on the idea of time shifts, all the attributes corresponding to a given note (*velocity*, *note ON* and *note OFF*) may be encoded at very distant positions of a sequence which could again stirs up some issues for learning systems.

2.2.2.3 *NoteTuple*

To alleviate this particular issue, the *NoteTuple* representation (Hawthorne et al., 2018) was recently proposed. In this method, each note is represented by a tuple composed by four attributes, namely, the *time offset* from the previous note, *pitch*, *velocity* and *duration*. The encoding of each attribute is categorical, with its own vocabulary instead of a large shared one (as in MIDI-like). As the time offset and duration vocabularies can potentially be very large, both are separated into *major* and *minor* tick fields. The time shift attribute counts 13 major and 77 minor ticks, representing 0 through 10 seconds and the duration attribute counts 25 major and 40 minor tick values. The result is a tuple containing six elements for each note. For notes played simultaneously, tuples are listed by order of increasing pitches as shown in the example Figure 2.2.

2.2.3 *Musical spaces*

One of the core research question in computer music remains to find an adequate representation for the relationships between musical objects. Indeed, the transcrip-



(a) Circular representation (b) K-nets (c) Tonnetz

Figure 2.3: Different representations of the pitch space which have been used for discovering algebraic patterns in music. (a) the circle representation of Marin Mersenne (Mersenne, 1972), (b) the K-nets discovered by Henry Klumpenhouwer (Klumpenhouwer, 1991) and (c) the tonnetz used in the Hexachord software (Bigo and Andreatta, 2017).

tion of music as symbols usually fails to provide information about harmonic or timbral relationships. In that sense, we can loosely say that the goal would be to find a target space, which could exhibit such properties between musical entities. Hence, finding representations of musical objects as spaces has witnessed a flourishing interest in the scientists community (Ukkonen, Lemström, and Mäkinen, 2003; Bigo, Giavitto, and Spicher, 2011; Typke, Veltkamp, and Wiering, 2004; Uitdenbogerd and Zobel, 1999). Many of the formalizations proposed over the past decades entail an algebraic nature that could allow to study combinatorial properties and classify musical structures. Here, we delimit a distinction between these methods into two types of representations: the rule-based and the agnostic approaches.

In the rule-based stream of research, several types of spaces have been developed since the Pythagoreans. Indeed, Marin Mersenne allowed to discover many algrebraic and geometric structure in classical music through his circular representation of the pitch space in the 17th century (Mersenne, 1972; Mesnage, 1997; Vieru, 1995). Many years later Henry Klumpenhouwer present a new space for representing music called the K-nets (Klumpenhouwer, 1991). This approach leaded to reveal some structural aspects in music through the many isographies of the networks (Lewin, 1994; Perle, 1996; Lewin, 1990). Finally, we can cite the well-known Tonnetz, invented by Euler in the 18th century (Euler, 1739) where the symbolic pitches are geometrically organized in an Euclidean space defined by infinite axes associated with particular musical intervals. We can see examples of these different spaces Figure 2.3.

In a mathematical point of view, all these representations are equivalent methods of formalizing the structural properties of the equal-tempered system (i.e. the division of the octave into twelve equal intervals). They provide a novel and powerful tool for the analysis of harmonic progression. Their combinatorics nature has aroused a lot of composition treatises and techniques which are valued for the pedagogical benefits they offered to students by transmitting knowledge in the manipulation of musical materials (Bigo et al., 2015). Moreover, these models have proven their ability to enhance human creativity, since they have been used in contemporary music for example in the composition of the so-called *Hamiltonian songs* (Bigo and Andreatta, 2014). These were created by following the 124 possible Hamiltonian cycles – which refer in graph theory to a path passing through all possible nodes and ending precisely where it start – present in the Tonnetz (Albini and Antonini, 2009).

There are two main benefits of this type of rule-based approach. First, once the model is built, it can be straightforward to analyze some of its properties (based on the defined sets of rules). Second, we can also understand the scope where the model should be efficient based on its construction. But as it is defined, a rule-based approach represent the particular vision of the designer that has thought and crafted the corresponding rule sets. Hence, the corresponding musical spaces will provide a given set of interactions. It is interesting to ask if we could develop a more empirical discovery of these spaces that could provide more generic musical relationships. These kinds of spaces could allow to exhibit properties in musical scores in a way that we never would have thought. In doing so, we could then find some new relevant features and metric relationships between musical entities and develop innovative applications. Hence, in the following, we consider that the important properties of a space are not necessarily its dimensions (like in the rule-based approaches), but rather the metric relationships or distances between objects inside this space (like in the agnostic approach that we seek to develop).

However, we remain conscious of the limitations of such agnostic spaces. Indeed, these are still indirectly the product of our design of the learning algorithms. Furthermore, they might be highly dependent on the dataset used for their construction. Finally, there might be no direct ways to analyze their properties nor prove their efficiency.

2.3 MACHINE LEARNING TOOLS

In the following section, we propose an overview of all the machine learning concepts that have been used during this thesis. First of all, we propose a formal de-

scription of what has been named *machine learning* by laying the necessary bases for an algorithm to be able to learn. Then, we detail the training procedure and the notions of artificial neurons and neural networks. Finally, we develop the architectures and specificities of all models and techniques useful for our work.

2.3.1 *Basics*

2.3.1.1 *Formal description*

Learning can be defined as the process of acquiring, modifying or reinforcing knowledge by discovering new facts and theories through observations (Gross, 2015). To succeed, learning algorithms need to grasp the generic properties of different types of objects by "observing" a large amount of examples. These observations are collected inside training datasets that supposedly contain a wide variety of examples. There exists three major types of learning :

- **Supervised learning:** Inferring a function from labeled training data. Every sample in the dataset is provided with a corresponding groundtruth label.
- **Unsupervised learning:** Trying to find hidden structure in unlabeled data. This leads to the important difference with supervised learning that correct input/output pairs are never presented. Moreover, there is no simple evaluation of the models accuracy.
- **Reinforcement learning:** Acting to maximize a notion of cumulative reward. This type of learning was inspired by behaviorist psychology. The model receives a positive reward if it outputs the right object and a negative one in the opposite.

In computer science, an example of a typical problem is to learn how to classify elements by observing a set of labeled examples. Hence, the final goal is to be able to find the class memberships of given objects (e.g. a sound played either by a piano, an oboe or a violin). Mathematically we can define the classification learning problem as follows.

Given a training dataset $\mathcal{X}$ of N samples $X = \{x_1, ..., x_N\}$ with $x_N \in \mathbb{R}^D$, we want assign to each sample a class inside the set $Y = \{y_1, ..., y_M\}$ of M classes with $y_M \in \{1, ..., M\}$.

To do so, we need to define a model Γ_θ that depends on the set of parameters $\theta \in \Theta$. This model can be seen as a *transform* mapping an input to a class value such that

$$\hat{y} = \Gamma_\theta(x_i) \tag{2–1}$$

In order to define the success of the algorithm, but also to allow learning, we further need to define a score function,

$$h = h_\theta : \mathbb{R}^{\mathcal{D}} \to \mathbb{R}^M \tag{2–2}$$

and a loss function,

$$\mathcal{L} = \mathcal{L}_{X,Y} : \Theta \to \mathbb{R} \tag{2–3}$$

The role of the score function is to determine the membership of a given sample x_i to a class y_m. In a statistical setting, we can interpret this function as the probability of belonging to a given class.

$$h_m(x_i) = p(\hat{y} = y_m \mid x_i, \theta) \tag{2–4}$$

Considering the basic rules of probability distributions, we have

$$\begin{cases} \sum_{m=1}^{M} h_m(x) = 1 \\ \\ h_m(x) \in [0,1], \qquad\qquad \forall x \in \chi, \quad \forall m = 1, ..., M \end{cases} \tag{2–5}$$

On the other hand, the loss function computes the difference between predictions of the model and groundtruths. In order to learn the most efficient model, we need to update its parameters θ by minimizing the value of this loss function

$$\hat{\theta} = argmin_{\theta \in \Theta}(\mathcal{L}_{X,Y}) \tag{2–6}$$

There is usually no analytical solution and sometimes not even a single minimum for this problem. The *optimal* set of parameters $\hat{\theta}$ is usually approximated by updating a sequence of θ_n that iteratively decreases the loss function. The *update rules* of the model describes how θ_{n+1} is obtained from θ_n. Other numerical parameters, called *hyper-parameters*, can influence the model but, unlike θ, they are not considered in the optimization problem.

Therefore, we rely on the Gradient Descent (GD) algorithm (Cauchy, 1847) to find a potential solution (minimum) to this problem by updating the parameters iteratively depending on the gradient of the error as defined as

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_{X,Y}(\theta_n) \tag{2–7}$$

The gradient of $\mathcal{L}_{X,Y}$ is associated with a vector in the parameter space θ which points out in the direction in which $\mathcal{L}_{X,Y}$ growths the most. Hence, geometrically speaking, the update rule corresponds to move each parameter θ in a direction

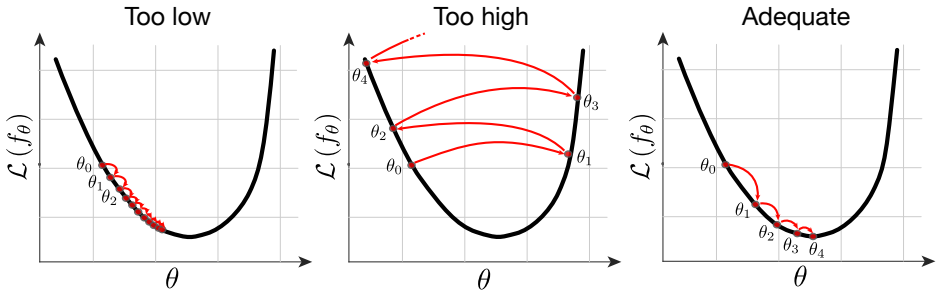


Figure 2.4: Impact of different values of the learning rate on the convergence of gradient descent

that lowers the value of the loss function by a step of the meta-parameter η, called the *learning rate*. The convergence highly depends on the value of η, as displayed in Figure 2.4. A convex loss with an adequate value η will provide a fast and easy convergence. Unfortunately, in most of the cases, the loss function is highly non-convex and possess several local minima. This does not imply any theoretical guarantees of the convergence to the global minimum.

In order to avoid the convergence on a local minimum, it is possible to consider the loss over a single example $\mathcal{L}_{x_i,y_i}$, and perform updates for each example separately. Using a single training sample $x(i)$ at each iteration of the GD process allows to cover more values of the parameters, and is called Stochastic Gradient Descent (SGD)

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_{x_i,y_i}(\theta_n) \tag{2–8}$$

Note that the GD algorithm actually corresponds to computing the mean of the SGD over the whole training dataset as the loss is defined as

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \mathcal{L}_{x_i,y_i}(\theta) \tag{2–9}$$

While GD must compute the derivatives of all the terms in the sum before updating the parameters (a costly operation when the number of samples is large), SGD can decrease the value of the loss, without computing all derivatives. However, as the gradient may take largely different values for each example, we usually rather rely on successive partitions $X_{j=\{1,\cdots,B\}}$ of the data set and apply the update rule

$$\theta_{n+1} = \theta_n - \eta \frac{\delta \mathcal{L}_{X_j,Y_j}}{\delta \theta}(\theta_n) \tag{2–10}$$

This process called Mini-Batch Gradient Descent (MSGD) allows to parallelize and thus speed up the computation of the GD algorithm.

2.3.1.2 *Training*

A single phase of training is referred to an *epoch*, which corresponds to one iteration of the training loop defined as listed in Algorithmus 1.

Data : X,Y,$\mathcal{L}_{X,Y}(\theta_n)$
Result : The parameters θ of the model Γ_θ minimize the value of the loss function $\mathcal{L}_{X,Y}(\theta_n)$
Random initialization of θ;
while $\hat{\theta} \neq argmin_{\theta \in \Theta}(\mathcal{L}_{X,Y})$ **do**
  Compute a prediction $\hat{y}$ for inputs x_i depending on the current parameters θ_n
  $\hat{y} = \Gamma_\theta(x_i)$;
  Evaluate the error by comparing the predictions groundtruth y_i with an arbitrary distance function F
  $\mathcal{L}_{X,Y} = F(\hat{y}, y_i)$;
  Update the parameters of the model in order to decrease the loss value, by relying on the derivatives of the error
  $\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_{X,Y}(\theta_n)$;
end

Algorithmus 1 : Training algorithm of a model Γ_θ for a classification task. X is the training set composed by a N samples $X = \{x_1, ..., x_N\}$, Y is the set of M classes that correspond with the samples $Y = \{y_1, ..., y_M$ with $y_i \in \{1, ..., M\}$

At this point, the question arises to know whether the number of epochs has to be the largest possible to have the most efficient model. Unfortunately, this is generally not the case due to the phenomenon known as over-fitting. Indeed, if the model learns "too precisely" the training dataset properties or if it counts too many parameters for the problem, it will learn the samples themselves and will not be able to generalize its learned concepts to unseen contexts anymore. This can be understood graphically by looking at Figure 2.5.

The first case on the left illustrate a model that has not been trained enough. Its classification function is too simple to separate efficiently the samples. In the opposite we can see on the center a model that has made too much training epoch or with an overly complex classification function. In this case, it will be very efficient for tasks on this particular dataset but very bad on other example. Finally, the right of the figure shows a model that has made the optimal number of epochs. Generally speaking, these questions relate to the notion of *generalization*, which

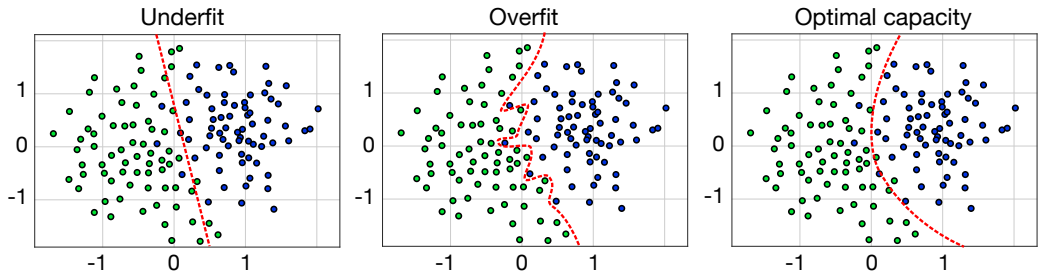


Figure 2.5: Depending on their complexities, models can tend to underfit (left), overfit (middle) or have an adequate capacity for the problem (right)

refers to the ability to accurately predict independent unknown test data.

To prevent the over-fitting, we split the data into three datasets, namely the *training set*, the *testing set*, and the *validation set*. We use the training dataset in order to update the parameters, and then we compute the loss with the test one. When the training loss value tends to zero (a model "knows" each sample, zero miss-predictions are made), the test loss will re-increase because of over-fitting as depicted in Figure 2.6. We stop the training at this point and assess our model on the validation dataset to get the expected final accuracy of the model on unknown data.

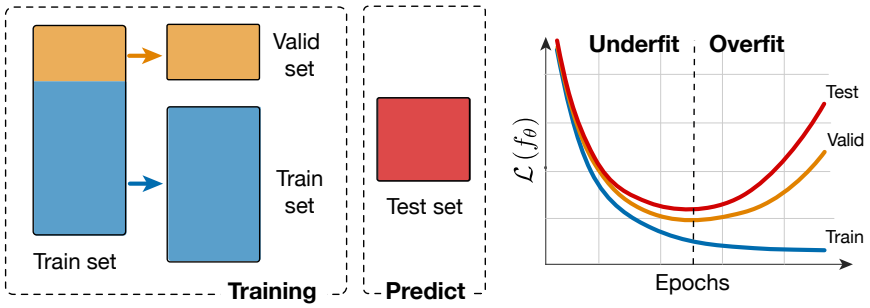


Figure 2.6: Separating the dataset in order to alleviate overfitting

Another solution to prevent over-fitting is to apply regularization to the model or the learning process. Examples of regularization include adding noise to the input or restrict the values of the parameters through weight decay. We can also add a regularization term to the loss in order to restrict the magnitude of the value that the parameters can take. The loss can then be written as

$$\mathcal{L}_{X,Y,\lambda}(\theta) = \mathcal{L}_{X,Y}(\theta) + \lambda\mathcal{R}(\theta) \tag{2–11}$$

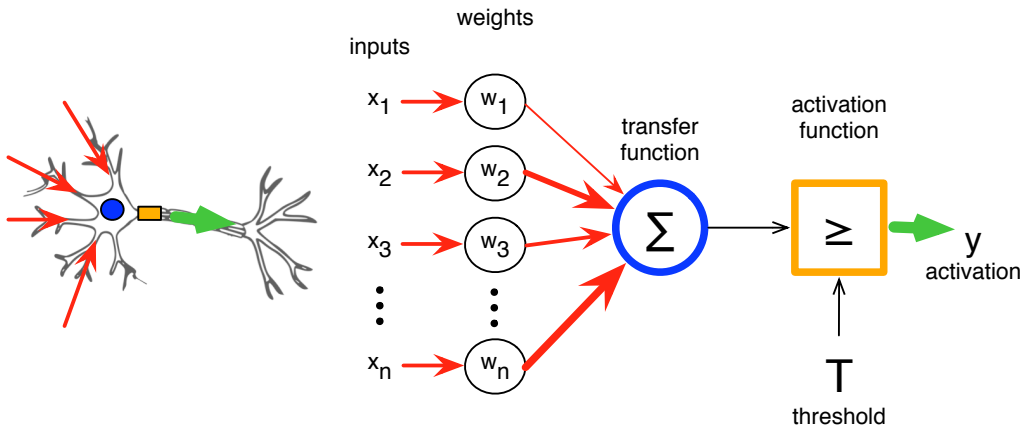


Figure 2.7: A biological neuron (left) is approximated through affine transform and activation function (right).

where $\mathcal{R}$ is usually a l_p-norm on the space of parameters and λ is a meta-parameter, which controls the strength of the regularization. When $\Theta \approx \mathbb{R}^K$, the l_p-norm is written as:

$$\|\theta\|_p = \left(\sum_{k=1}^{K} |\theta_k|^p \right)^{\frac{1}{p}} \tag{2–12}$$

Most of the time, the l_2 or l_1 norms are used. While l_1-norm regularization leads to sparse parameters, where only a few have a "significant" value and the remaining ones are very close to zero, l_2-norm regularization leads to uniform "low" value parameters.

2.3.1.3 *Artificial neurons*

Now that we have a global overview of learning algorithms, we will define the model Γ_θ as a neural networks. A wide part of research in machine learning has focused on the concept of artificial neural networks. Indeed, the most widely known basis of intelligent behavior as we know it, is the biological neuron. Therefore, scientists tried to mimic its mechanisms, tracing back to the original model proposed in McCulloch and Pitts, 1943. An artificial neuron is composed of multiple weights and a threshold, that together define its parameters, as depicted in Figure 2.7.

The activation of the output of the neuron will be governed by an affine transform and a non-linear activation function. Mathematically, with N inputs a neuron output is defined as

$$y = \sum_{i=1}^{N} x_i w_i + T \tag{2–13}$$

with w_i the learned weights and T the threshold of activation. If we interpret this equation geometrically, we can see that it corresponds to an N-dimensional hyperplane. Therefore, a neuron can divide a space (akin to binary classification), or approximate a function (as the sum will give an output whatever X comes in). By organizing these neurons as layers where each neuron (also called units) transforms the input independently, we obtain the perceptron. These layers are then stacked one after another, where the input of a layer is the output of the previous one, to obtain the well-known multi-layer perceptron. The number of layers is called the depth of the model. Hence, the multi-layer neural network allows combining non-linear activations in order to process more complex tasks.

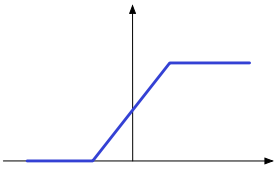
However, a problem arise from this representation. Indeed, to update the parameters with gradient descent method (Equation 2–7) the output space has to be continuous (e. g., the value of y must be continuous to be differentiable, $y(x)$ is of class $\mathcal{C}^1$). To alleviate this problem we need to use different activation function denoted as $\sigma(x)$, and we now consider T as a bias denoted as b, leading to

$$y = \sigma(\sum x_i w_i + b) \tag{2–14}$$

Three examples of common activation functions are showed in the following.

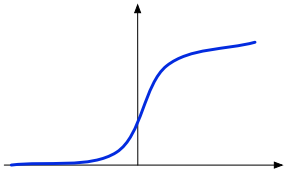
- Piecewise linear :

$$\sigma(x) = \begin{cases} 0 & \forall\ x \leqslant x_{min} \\ mx + b & \forall\ x_{max} > x > x_{min} \\ 1 & \forall\ x \geqslant x_{max} \end{cases}$$



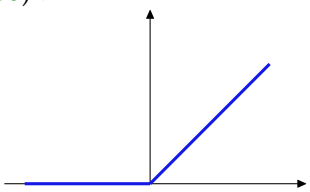
- Sigmoid :

$$\sigma(x) = \frac{e^x}{1 + e^{-\beta x}}$$



- Rectified Linear Unit (ReLU) (Hahnloser et al., 2000) :

$$\sigma(x) = \max(0, x)$$



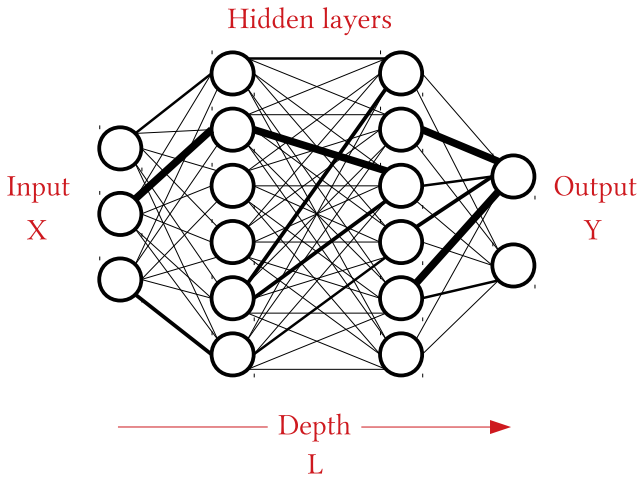


Figure 2.8: A fully-connected network with four layers. Units between two adjacent layers are fully pairwise connected.

2.3.1.4 *Neural networks*

We illustrate one of the most common architecture in Figure 2.8, the fully-connected network whose units between two adjacent layers are fully pairwise connected. This type of architecture is called feed-forward as the information moves in only one direction, forward, from the inputs to the outputs. There is no loops, backward connections or connections among units in the same layer. Moreover, the middle layers have no link with the external world, and hence are called hidden layers (Leshno et al., 1993). Here, we denote L the *depth* (i.e. the number of layers) of the network, and $y^l \in \mathbb{R}^{N_l}$ the output value of the l-th layer, where N_l is the number of neurons contained in this layer. For $1 \leqslant l \leqslant L$, the parameters of a layer are defined by a weight matrix $W^l \in \mathbb{R}^{N_l \times N_{l-1}}$ and a bias vector $b^l \in \mathbb{R}^{N_l}$. Therefore, the activation of neuron i in layer l is computed with the following equation:

$$h_i^l = \sigma\Big(\sum_{j=1}^{N_{l-1}} (W_{i,j}^l . h_j^{l-1}) + b_i^l\Big), \tag{2–15}$$

where $h_i^0 = x_i$. Moreover, as $b_i^l = W_{i,0}^l . h_0^{l-1}$, the previous equation now become

$$h_i^l = \sigma\Big(\sum_{j=0}^{N_{l-1}} (W_{i,j}^l . h_j^{l-1})\Big) \tag{2–16}$$

Finally, the output of the network is defined by $y_i = h_i^L$.

2.3.1.5 *Back-propagation*

Here, we can argue that the training of neural networks with a deep architecture (with a very large depth L) is nearly out of the question in light of the complexity of the computed function. However, this is without taking into account the method proposed by Rumelhart, Hinton, and Williams, 1986 and called *back-propagation*. This approach allows to decompose this procedure into a set of simple functions considering that adding layers is equivalent to adding a function computed on the previous output. Subsequently, in the *forward pass*, the output is updated depending on the parameters of the neurons in its layer. Thus, when attempting to acquire the contribution of a given neuron to the final error rate, we can apply the *chain rule* of derivations, to isolate its contribution inside the network. The error values acquired after a forward pass can be just propagated in reverse, starting from the output, and assessing the derivative of every neuron output given its parameters. This technique is applied recursively until all the weights of the network have been updated as depicted in Figure 2.9.

The forward pass allows to compute the loss $\mathcal{L}$ between the desired output y_i and the output of the network h_i^L. By using Equation 2–15, we define $\mathcal{L}$ on the output of layer $L-1$, the weight matrix W^L and σ the activation function (see Equation 2–17). Thus, we can define the total loss as an equation containing only the set of weight of the models, the inputs x_j and the desired output

$$\mathcal{L}(y_i, h_i^L) = \mathcal{L}(y_i, \sigma(\sum_{j=0}^{N_{L-1}} W_{i,j}^L . h_j^{L-1})) \tag{2–17}$$

$$= \mathcal{L}(y_i, \sigma(\sum_{j=0}^{N_{L-1}} W_{i,j}^L [...] \sigma(\sum_{j=0}^{N_0} W_{i,j}^1 . x_j))) \tag{2–18}$$

We denote e_i^l the derivative of the error for neuron i of layer l, and the output of layer $l-1$ before its activation function by

$$o_i^l = \sum_{j=0}^{N_{l-1}} W_{i,j}^l . h_j^{l-1}$$

The error for the last layer is defined based on the loss function

$$e_i^L = \frac{\delta \mathcal{L}(y_i, h_i^L)}{\delta h_i^L} \tag{2–19}$$

The chain rule allows to compute the errors e_i^l for all other layers of the NN. First, the chain rule is used to compute the derivative of $\mathcal{L}(y_i, h_i^L)$ with respect to

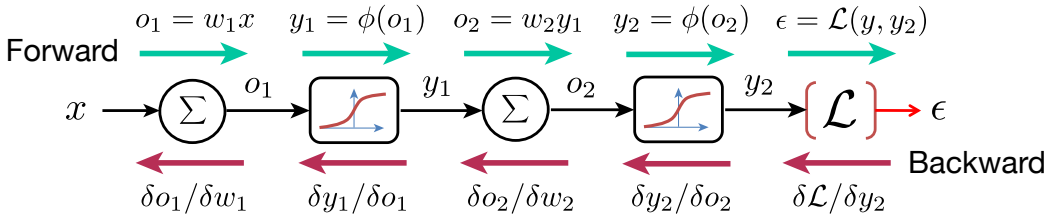


Figure 2.9: Computational graph for the forward and backward pass

δo_i^L. In Equation 2–20, the first term corresponds to e_i^L, whereas the second term is the derivative of the activation function.

$$\frac{\delta \mathcal{L}(y_i, h_i^L)}{\delta o_i^L} = \frac{\delta \mathcal{L}(y_i, h_i^L)}{\delta h_i^L} * \frac{\delta h_i^L}{\delta o_i^L} \tag{2–20}$$

$$= e_i^L * \sigma'(o_i^l) \tag{2–21}$$

Second, the chain rule is used another time to compute the derivative of $\mathcal{L}(y_i, h_i^L)$ with respect to the contributions of neurons in the previous layer h_i^{L-1}.

$$\frac{\delta \mathcal{L}(y_i, h_i^L)}{\delta o_i^L} * \frac{\delta o_i^L}{\delta h_i^{L-1}} = \frac{\delta \mathcal{L}(y_i, h_i^L)}{\delta o_i^L} * W_{i,j}^L \tag{2–22}$$

$$= e_i^L * \sigma'(o_i^l) * W_{i,j}^L \tag{2–23}$$

Thus, by taking into account all the connections emanating from the last layer

$$e_i^{L-1} = \sigma'(o_i^l) \sum_{j=0}^{M_L} W_{i,j}^L . e_j^L \tag{2–24}$$

Therefore, we obtain a relationship between the errors of two successive layers. This error can therefore be back propagated throughout the network, thanks to the generalization of Equation 2–24

$$e_i^l = \sigma'(o_i^l) \sum_{j=0}^{M_{l+1}} W_{i,j}^{l+1} . e_j^{l+1} \quad ; \quad l \in [1, ..., L-1] \tag{2–25}$$

Finally, the weights are updated in each layer by an amount proportional to the derivative of the error with respect to the associated weight

$$W_{ij}^l = W_{ij}^l + \eta . e_i^l h_j^{l-1} \tag{2–26}$$

Where η is the learning rate.

2.3.2 *Specific tools*

Other type of network based on different connections or operations have been developed. We now present the different specific architectures that were useful for our work, namely the CNN, the RNN, the attention mechanism, the Auto-Encoder (AE) framework, and finally the VAE.

2.3.2.1 *CNN*

Convolutional Neural Networks were inspired by the models of the visual system's structure proposed in Hubel and Wiesel, 1962. It is a category of neural networks that have proven very effective in areas such as image recognition and classification (Krizhevsky, Sutskever, and Hinton, 2012; Simonyan and Zisserman, 2014a; Simonyan and Zisserman, 2014b). There are four main operations in this kind of network, each processed by a different layer (see Figure 2.10).

CONVOLUTION The first layer is the convolution operator. Its primary purpose is to extract features from the input matrix. Indeed, each units $k \in \mathbb{N}$ in this layer can be seen as a small filter determined by the weights W_k and the bias b_k that we convolve across the width and height of the input data x. Hence, this layer will produce a 2-dimensional activation map h^k, that gives the activation of that filter across every spatial position

$$h^k_{ij} = (W^k * x)_{ij} + b_k \tag{2–27}$$

With the discrete convolution for a 2D signal defined as

$$f[m,n] * g[m,n] = \sum_{u=-\inf}^{\infty} \sum_{v=-\inf}^{\infty} f[u,v] * g[m-u, n-v] \tag{2–28}$$

The responses across different regions of space are called the receptive fields. During the training process, the network will need to learn filters that can activate when they see some recurring features such as edges or other simple shapes. By stacking convolutional layers, the features in the upper layers can be considered as higher-level abstraction such as composed shapes.

NON-LINEARITY As discussed in the previous section, we have to introduce non-linearities (NL) in the network in order to model complex relationships. Hence, before stacking every feature maps in order to obtain the output activations we apply a non-linear function like those introduced previously.

POOLING Spatial pooling (also called subsampling or downsampling) allows to reduce the dimensionality of each feature map. The principle behind the pooling

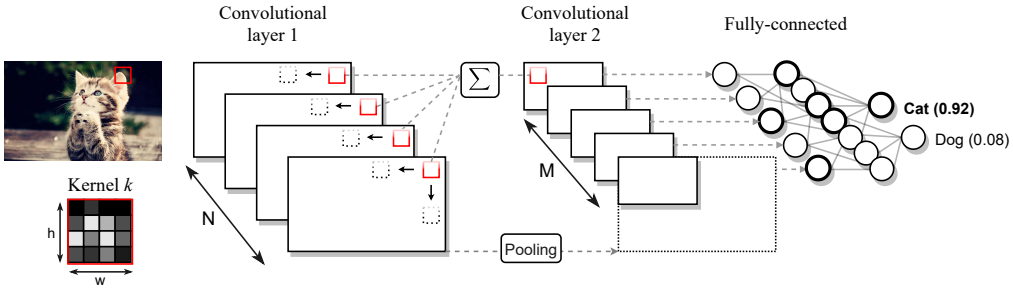


Figure 2.10: Convolutional neural network with two convolutional layers separated by a pooling layer and followed by a fully-connected network for classification.

operation is to define a spatial neighborhood (such as a $3 * 3$ window) and take the largest elements (max-pooling) or the average (average-pooling) of all elements in that window. In that way, we progressively reduce the spatial size of the input representation and make it more manageable.

FULLY-CONNECTED LAYER Based on the highest level features in the network, we can use these to classify the input into various categories. One of the simplest way to do that is to add several fully-connected layers (see Figure 2.8). By relying on this architecture, the CNN will take into account the combinations of features similarly to the multi-layer perceptron.

2.3.2.2 *Dense and Residual CNN*

We have seen that by stacking layers, the CNN was able to handle more and more complex shapes and thus increase its level of abstraction until it could identify specific objects. Theoretically, an extremely deep network is therefore capable of dealing with a maximum amount of detail, enabling it to perform highly demanding tasks such as facial recognition. Unfortunately, it has been shown in He et al., 2016 that, in practice, the norm of the gradient of the error decreases with each layer until it becomes insignificant enough to no longer allow the optimization of the network parameters creating a maximum threshold for depth with classical CNN. To alleviate this issue, the authors have proposed a novel architecture relying on residual connections between convolutional layers which add the outputs from previous layers to the outputs of stacked layers as depicted in Figure 2.11. Thus, the output of a layer in a residual network can be formally defined as

$$y = \mathcal{F}(x, \{W_i\}) + W_s x \tag{2–29}$$

where x and y represent the input and output of the layer respectively, $\mathcal{F}(x, \{W_i\})$ is the residual mapping to be learned and W_s is a linear projection applied to match the dimensions of x and y when the number of channels differs from one

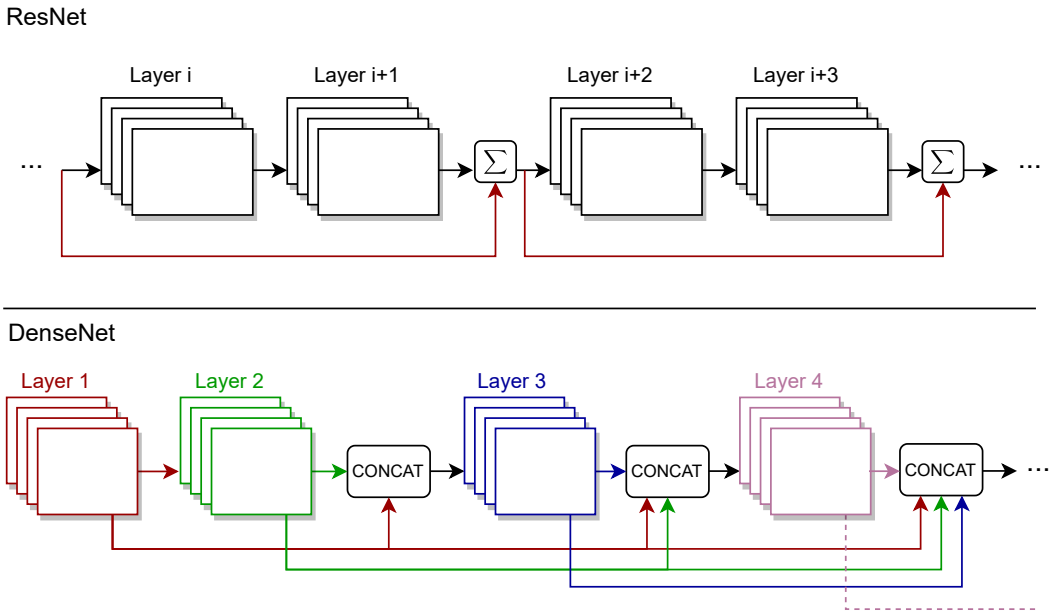


Figure 2.11: (Top) Residual CNN where the red lines represent the skip connection between layers (He et al., 2016). (Bottom) Densely connected CNN where all layers are connected with each other (Huang et al., 2017).

layer to another. These skip connections have shown to be highly valuable to allow the training of very depth networks and have enabled the residual networks to largely outperforms all the previous models in many visual recognition tasks.

Driven by this success, a multitude of architectures have emerged (Srivastava, Greff, and Schmidhuber, 2015; Huang et al., 2016; Larsson, Maire, and Shakhnarovich, 2016), including a particularly successful one called *DenseNet* (Huang et al., 2017). In this proposal, all layers are connected with each other to maximize the information flow between them leading to an increase in the number of direct connections between layers from L in a traditional architecture to $\frac{L(L+1)}{2}$. Besides, as we can see in Figure 2.11, the features are not combined by summation as for the residual network but by concatenation. Note that in this case, the matching of the features map size is mandatory. This model has been shown to foster the reuse of the learned features throughout the layers and thus to decrease the redundancy in the information flow resulting in the achievements of state-of-the-art performances in all the competitive visual tasks with fewer parameters.

2.3.2.3 *RNN*

Traditional neural networks do not allow information to persist in time. In other words, it could not use features extracted from previous events to inform later

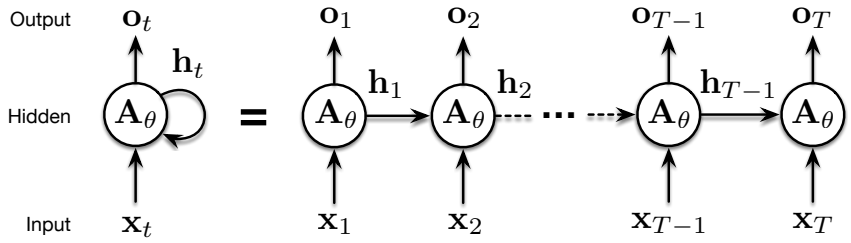


Figure 2.12: Loops of a RNN unfolded through time.

ones. To address this issue, specific networks with loop connections have been developed, called RNN (Elman, 1990). The idea is that a neuron now contains a loop to itself, allowing to carry information from one time step to the next. Hence, its activation can be defined as

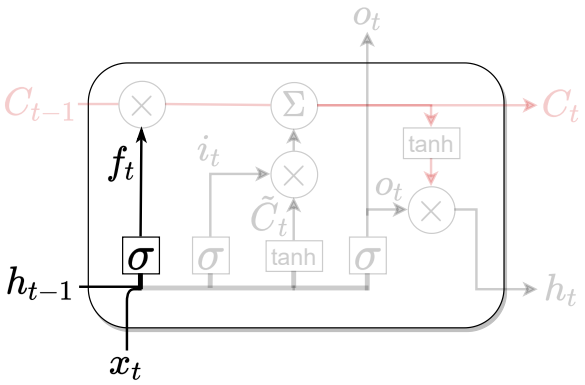
$$h_t = A(x_t, h_{t-1}) \tag{2–30}$$

With x_t the input and h_{t-1} the activation of the previous time step. By "unfolding" the networks, it can be thought of as multiple copies of the same feed-forward network, each passing a message to its successor (see Figure 2.12).

This chain-like nature underlines the intimate connection that RNN share to sequential events. Unfortunately, this kind of network is usually not able to learn long-term dependencies and are usually bound to succeed in tasks with very short contexts. This problem was explored in depth in Bengio, Simard, and Frasconi, 1994, exhibiting the theoretical reasons behind these difficulties, such as the vanishing or exploding gradient. To alleviate these issues, Hochreiter and Schmidhuber introduced the LSTM network (Hochreiter and Schmidhuber, 1997). These cells also have this chain-like structure, but the repeating module has a different structure. Indeed, the key element of a LSTM network is a signal called the cell state which runs straight down the entire structure. This signal can be seen as the main information that is passed from an "unrolled units' to another at each time steps. In the following, we denote the cell state at the time step t as C_t. During its crossing, this information can be modified or not or even totally forgotten depending of the input of the network x_t and the output of the previous units h_{t-1}. This is the role of four other elements in the structure called gate defined with weights W and bias b that we now present in details.

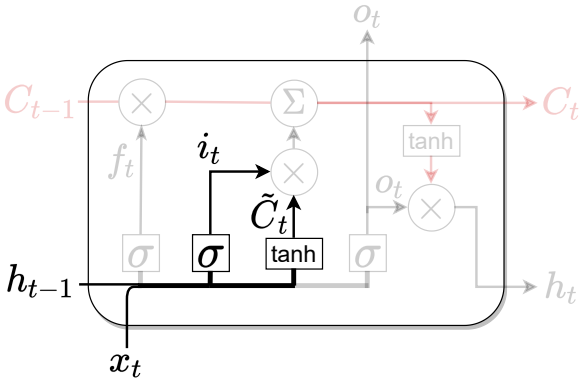
1. **The forget gate** is a simple sigmoid layer which is here to decide if we keep the previous information in the cell state or not. It takes into account h_{t-1} and x_t, and outputs a number f_t between 0 and 1 where 0 represents "com-

pletely forget this" and 1 represents "completely keep this".



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

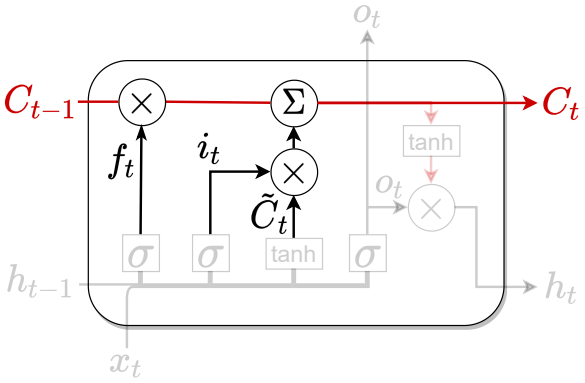
2. **The input gate** allows to decide what new information we are going to store in the cell state. This as two parts, a sigmoid layer dedicated to decide which values will be updated or not depending on i_t and a tanh layer which creates a vector of new candidate values $\tilde{C}_t$ for the update.



$$i_t = \sigma(W_f[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_f[h_{t-1}, x_t] + b_C)$$

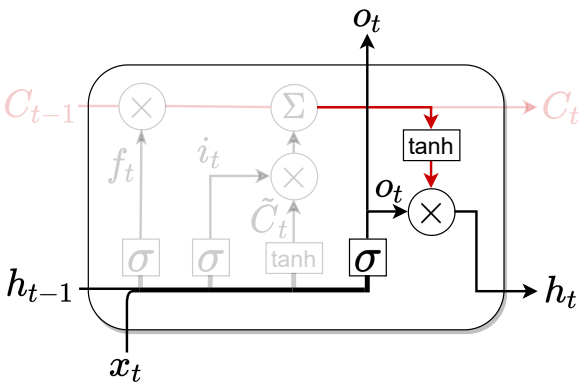
3. **The update gate** that its goal is to actually do what we decided before. Therefore we multiply the old state by f_t and add it $i_t * \tilde{C}_t$. The resulting cell state C_t is passed to the next time step unit.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

4. **The output gate** finally decide what the network is going to output depending of the input x_t and of the cell state C_t. The combination of a sigmoid

layer and a tanh layer are applied to achieve this.



$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

2.3.2.4 *Auto-Encoders*

One of the most successful type of models that has been recently proposed in the machine learning field is the AE. These networks are composed of an encoder $\mathcal{E}_\phi(x) : \mathbb{R}^x \rightarrow \mathbb{R}^z$, which embeds the input data in a lower dimensional space $\mathbb{R}^z$ where the second part, the decoder $\mathcal{D}_\theta(z) : \mathbb{R}^z \rightarrow \mathbb{R}^x$, tries to reconstruct the original input from this code so that $\hat{\mathbf{x}} = \mathcal{E}_\phi(\mathcal{D}_\theta(x)) \approx \mathbf{x}$ as depicted in Figure 2.13.

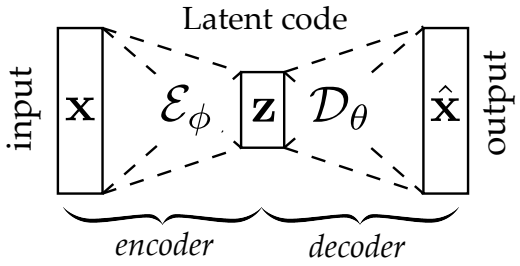


Figure 2.13: Auto-encoder architecture. $\mathbf{x}$ is *encoded* into a latent space $\mathbf{z}$. The *decoder* aims to reconstruct an estimation $\hat{\mathbf{x}}$ from $\mathbf{z}$

Then, the reconstructed output are compared to the input through a loss function (typically a Mean Squared Error (MSE)) which the network try to minimize

$$\mathcal{L}(\hat{\mathbf{x}}, \mathbf{x}) = \| e(d(\mathbf{x})) - \mathbf{x} \|^2 \tag{2–31}$$

In order to avoid learning the identity function, the encoded (*latent*) space is chosen to have a much lower dimensionality than the input space. By doing so, the model learns to compress effectively the input in the final latent space.

2.3.2.5 *Variational Auto-Encoders*

A very successful improvement of the AEs, called the Variational Auto Encoder (VAE), have been proposed in Kingma and Welling, 2013. This class of model is based on the *Variational Inference* paradigm which aims to approximate a conditional density $p(\mathbf{z} \mid \mathbf{x})$ of latent variables $\mathbf{z}$ given observed ones $\mathbf{x}$ by optimising a family of densities over the latent variables (Blei, Kucukelbir, and McAuliffe, 2017). Indeed, a data distribution can be defined with the marginal distribution $p(\mathbf{z})$ and the probability of generating $\mathbf{x}$ with given latent variable $\mathbf{z}$ as following

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z} \tag{2–32}$$

In this equation, both distribution ($p(\mathbf{x} \mid \mathbf{z})$ and $p(\mathbf{z})$) are usually unsolvable through a closed analytical form. However, the Variational Inference allows the approximation of $p(\mathbf{z} \mid \mathbf{x})$ by a simpler distribution $q(\mathbf{z} \mid \mathbf{x}) \in \mathcal{Q}$ where $\mathcal{Q}$ is a parametrized family of distribution. In order to assess the quality of this approximation, we measure its difference with the posterior distribution by means of the KullBack-Leibler Divergence ($\mathcal{D}_{KL}$) defined by:

$$\mathcal{D}_{KL}[q(\mathbf{z} \mid \mathbf{x}) || p(\mathbf{z} \mid \mathbf{x})] = \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z} \mid \mathbf{x})] \tag{2–33}$$

By applying the Bayes' rule on the previous equation, we thus obtain

$$\mathcal{D}_{KL}\big[q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z} \mid \mathbf{x})\big] = \mathbb{E}_{q(\mathbf{z})}\big[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{x} \mid \mathbf{z}) - \log p(\mathbf{z}) + \log p(\mathbf{x})\big] \tag{2–34}$$

Here, we can notice that $p(\mathbf{x})$ and $q(\mathbf{z})$ are mutually independent allowing the reformulation of Equation 2–34 as

$$\log p(\mathbf{x}) - \mathcal{D}_{KL}\big[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})\big] = \mathbb{E}_{\mathbf{z}}\big[\log p(\mathbf{x} \mid \mathbf{z})\big] - \mathcal{D}_{KL}\big[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})\big] \tag{2–35}$$

Therefore, the aim of the VAE is to optimize the divergence of the two distributions. To do so, it rely on its *encoder* and its *decoder* defined respectively as parametric functions $q_\phi(\mathbf{z})$ with $\phi \in \Phi$ and $p_\theta(\mathbf{z})$ with $\theta \in \Theta$. Assuming $\log p(\mathbf{x})$ is a constant value, we can write the final optimization problem as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z})}\big[\log p_\theta(\mathbf{x} \mid \mathbf{z})\big]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{KL}\big[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z})\big]}_{\text{regularisation}} \tag{2–36}$$

As we can see, this loss function is composed of two different terms. The first term is the *reconstruction loss* (or expected negative log-likelihood), which encourages to learn an accurate reconstruction of the data. On the other hand, the Kullback-Leibler Divergence act as a *regularizer* as it measures how much information is lost by relying on approximate $q_\phi(\mathbf{z} \mid \mathbf{x})$ instead of the true latent distribution $p_\theta(\mathbf{z})$. Moreover, forcing the latent distribution of the data to be close to an isotropic

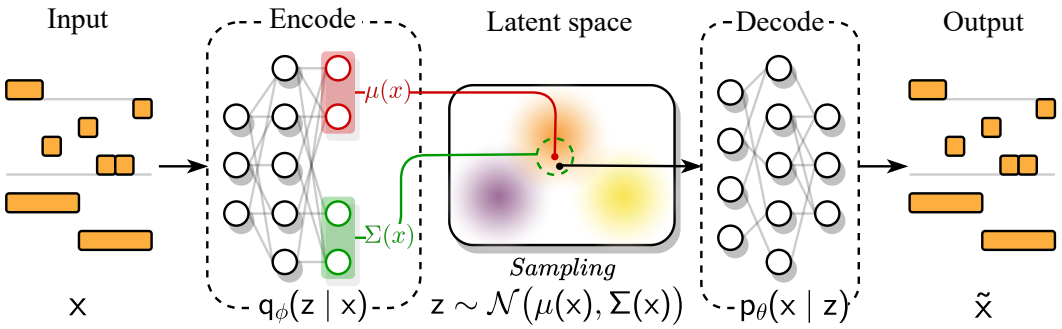


Figure 2.14: Architecture of a VAE with sampling in the latent space where the data are smoothly organized thanks to the regularisation.

normal distribution (choosing a prior $p(\mathbf{z})$ with mean zero and variance one $p(\mathbf{z}) \sim \mathcal{N}(0, 1)$) avoids the network to project each inputs in a very different region of the latent space, hence favoring close embedding vectors for similar inputs (in other words, they will be close together in the final space).
Therefore, the encoder outputs the pair $\left[\mu(\mathbf{x}), \Sigma(\mathbf{x})\right]$, which are parameters of the multivariate Gaussian probability density $q_\phi(\mathbf{z} \mid \mathbf{x})$ and the decoder is then fed with sample from this density and outputs a reconstruction $\tilde{\mathbf{x}}$ as depicted Figure 2.14

REPARAMETERIZATION TRICK As the training of the VAE is obtained through gradient descent, the sampling operation from the latent space would render the estimation of the gradient extremely noisy which can hinder the convergence. The *reparameterization* trick tackles this particular issue (Kingma and Welling, 2013). The main idea is to move the sampling operation outside of the network definition to ensure that it remains trainable, as depicted in Figure 2.15. If we define $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$ and standardize it so that $\bar{\mathbf{x}} \sim \mathcal{N}(0, 1)$, then we can revert the standardization by simply computing $\mathbf{x} = \mu + \Sigma^{\frac{1}{2}}\bar{\mathbf{x}}$. Therefore, we can perform the overall sampling operation, by first sampling from a standard normal distribution with $\epsilon \sim \mathcal{N}(0, 1)$ and then convert it to the desired Gaussian with a specific mean and variance $\mathbf{z} = \mu(\mathbf{x}) + \Sigma^{\frac{1}{2}}(\mathbf{x})\epsilon$. This allows to perform the sampling process outside of the network (eg. without any dependency to the network parameters). This means that the stochastic component of the gradient will not be taken into account for the gradient computation.

The *reparameterization trick* provides an easy way of computing gradients of the form $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)]$ for many common densities $q_\phi(z)$. The key idea is to express a sample from $q_\phi(z)$ as a function of a sample ϵ from some fixed distribution $p(\epsilon)$.

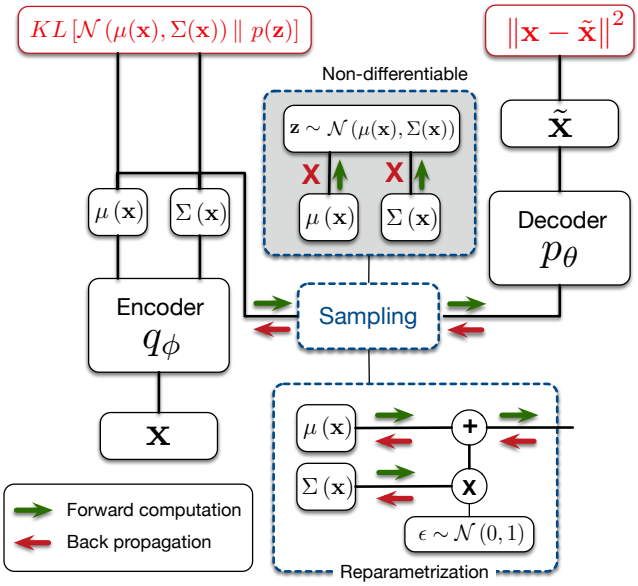


Figure 2.15: The reparameterization trick

2.3.2.6 *Attention mechanism*

When we process sequential information, such as reading a book, a score or watching a video, the amount of information that reaches our brain is enormous and not all of it is equally important in interpreting a given situation. Therefore, we have learned to unconsciously select from this flow the most relevant ones. For example, if we want to find the tonality of a given excerpt of music by reading its score, we will focus on particular notes and accidentals while not paying attention to others. In broader terms, we are able to assess the *interdependence* between input data and a desired output.

The same idea has been adapted to machine learning systems leading to the so-called AM (Bahdanau, Cho, and Bengio, 2014) which have been proposed in a wide variety of machine learning algorithms such as text translation (Firat, Cho, and Bengio, 2016), image captioning (Xu et al., 2015) or sentence summarization (Rush, Chopra, and Weston, 2015). Its original use was intended to discriminate salient elements inside sequential data. Hence, given a sequence of n inputs $\{x_0, ..., x_{n-1}\}$ and a context C, the attention mechanism computes a n-dimensional vector A of weights reflecting the relevance of each x_i in the context C. By computing the dot product $x \cdot A$, we finally obtain a weighted arithmetic mean of the input.

In a more computer science oriented way, we can defined the AM as a function which aims to map a *query* and a set of *key-value* pairs to an output. The keys ($\mathbf{K}$) correspond to the weights describing the relative relevance of the inputs in a

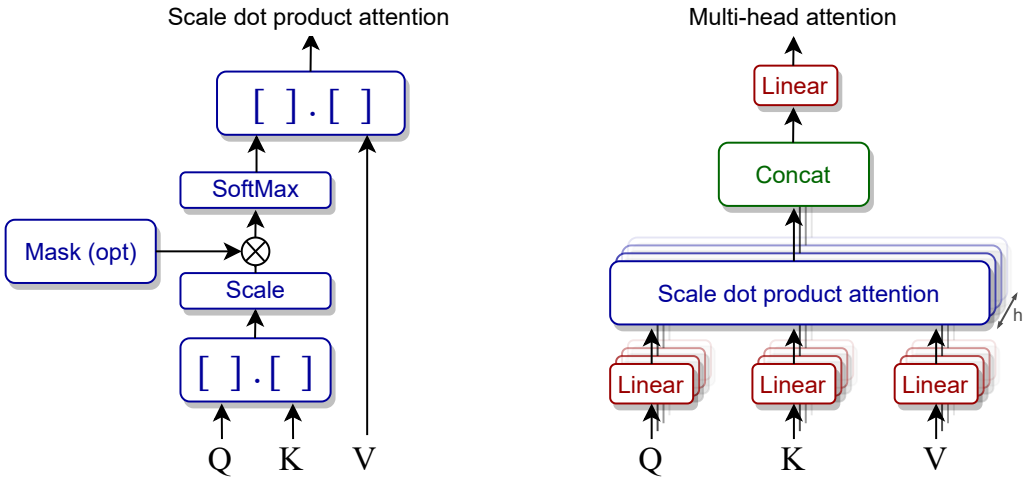


Figure 2.16: (Left) Scale dot product attention mechanism. (Right) Multi-head attention mechanism

sequence for a given query (**Q**) and the values (**V**) result from the multiplication of the weights with the input sequence. This formalism has led to two major extended versions of the AM, namely the *Scaled Dot-Product Attention* and the *Multi-Head Attention* (Vaswani et al., 2017).

For the first one, the matrix of outputs is computed as

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^\top}{\sqrt{d_k}})V \tag{2–37}$$

where d_k is the dimension of the keys and queries (see Figure 2.16). The main improvement of this algorithm appears when dealing with large values of d_k. Indeed, the scale factor $\frac{1}{\sqrt{d_k}}$ prevents the magnitude of the dot product from overgrowing which causes the gradient of the softmax function to vanish.

For the Multi-Head Attention, the authors have gone further than performing a single attention function with d_{model}-dimensional K,V and Q. They rely instead on h different learned linear projections to d_k dimensions for the queries and the keys and to d_v dimensions for the values. Then the attention function is applied in parallel to all the projected version of K,V, and Q leading to d_v-dimensional output values. Finally, these are concatenated and linearly projected to obtain the final output as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2–38}$$

where

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2–39}$$

with

$$\begin{cases} W_i^Q \in \mathbb{R}^{d_{model} \times d_k} \\ W_i^K \in \mathbb{R}^{d_{model} \times d_k} \\ W_i^V \in \mathbb{R}^{d_{model} \times d_v} \\ W^O \in \mathbb{R}^{hd_v \times d_{model}} \end{cases}$$

where W_i^Q, W_i^K, W_i^V and W^O are the projections of Q, K, V and the output O respectively. The whole algorithm is depicted Figure 2.16.

As the title – *Attention is all you need* – of this paper suggests, these algorithms are so efficient that a model based entirely on attention has outperformed the state-of-the-art recurrent architectures on several tasks.

2.4 EMBEDDING SPACES

2.4.1 *Apparition and formalism*

An embedding space is considered in our context as a space of lower dimensionality which can be found from the high-dimensionality space of the input. Then, inside this space, the embedding of an object will be a representation of this object inside the lower-dimensionality space in such a way that some targeted algebraic properties are preserved. From a topological point of view, one space X is said to be embedded in another space Y when the properties of Y restricted to X are the same as the properties of X. However, in our case, we want to target certain properties of similarities between the different object inputs, such that the distances inside the embedded space mimics the targeted relationships. For our problem, this amounts to find a meaningful representation for musical elements inside a space, in which the distance between the musical entities would faithfully represent their musical similarity. It means that we seek a transform that could map every notes and chords to these low-dimensional vectors, for which the distance between vectors would carry semantic relations. An example of embedding space is depicted in Figure 2.17. For example, in this space, the distance between the word embedding vector for "strong" and the one for "stronger" is the same than between "clear" and "clearer". We can see that even though the dimensions of this space do not have a particular meaning, the metric relationships inside this space do mirror some semantic meaning.

During the last decades, most of the work devoted to embedding spaces has been centered on NLP through word embedding space models. Indeed, in 2003

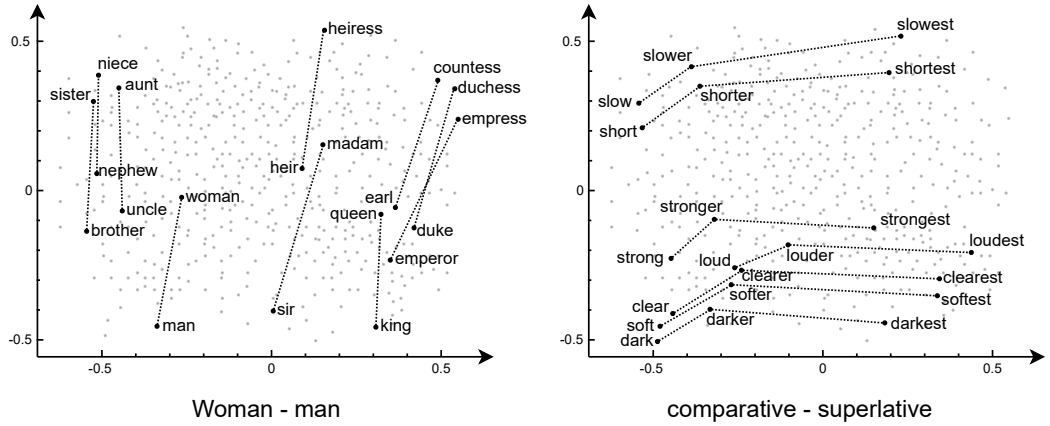


Figure 2.17: Set of visualizations that show interesting patterns relying to the vectors differences between related words in an embedding space learned with GloVe ('GloVe: Global Vectors for Word Representation').

Bengio et al. used for the first time a word embedding inside a neural language model (Bengio et al., 2003). Following this seminal work, many word embedding algorithms were developed including the well known Latent Semantic Analysis (LSA) (Landauer, 2006), the Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan, 2003) and the model proposed in Collobert and Weston, 2008, which altogether form the foundation for most of the current approaches. Since then, several NLP tasks such as automatic caption generation (Reed et al., 2016; Ren et al., 2016; Vinyals et al., 2015), text classification based on sentiments (Kiros et al., 2015) or speech recognition (Mroueh, Marcheret, and Goel, 2015; Noda et al., 2015), include the computation of an embedding space for words as their first step to implement more complex behaviors.

Embedding spaces can be learned through machine learning techniques. Indeed, we saw in the previous section that neural network transform an input in order to minimize the value of the loss function. Hence, we can manage to learn a transform that provide a mapping of each samples in a continuous N-dimensional space, that carry information on relationships between elements.

2.4.2 *Successful models*

In the following, we will present the currently best-performing models namely *Word2vec* (Mikolov et al., 2013a; Mikolov et al., 2013b), and Global Vectors for word representation (GloVe) (Pennington, Socher, and Manning, 2014) that provides state-of-the-art results for word embeddings. We will consider that the learning algorithm is fed with sentences composed of words such that $s = \{x_1, ..., x_n\}$. In

that case, a word w_t is said to be in a context $c = \{w_{t-p}, ..., w_{t-1}, w_{t+1}, ..., w_{t+p}\}$. We will talk about the past context of w_t as $\{w_1, ..., w_{t-1}\}$ and the future context as $\{w_{t+1}, ..., w_{t+n}\}$.

2.4.2.1 *Word2vec*

There are two critical aspects of learning embedding spaces that allows to produce interesting embeddings and also evaluate their quality. First, the training objective of the corresponding learning algorithms should make it effective for encoding general semantic relationships. Second, the computational complexity of such an objective should be low for this task, while providing an efficient coding scheme. Hence, the idea is to learn a model $f(w_t, ..., w_{t-n+1}) = \hat{P}(w_t \mid w_1^{t-1})$ that is decomposed in two parts. First, a mapping C from any words to a real vector $C(i) \in \mathbb{R}^m$ that represent the distributed feature vectors. Then, the probability function over words expressed with C through a function g which maps an input sequence of feature vectors $\{C(w_{t-n+1}), ..., C(w_{t-1})\}$ to a conditional probability distribution over words for the next word w_t. The i-th element of the output of g determines the probability $\hat{P}(w_t \mid w_1^{t-1})$ (Bengio et al., 2003). This architecture is depicted Figure 2.18.

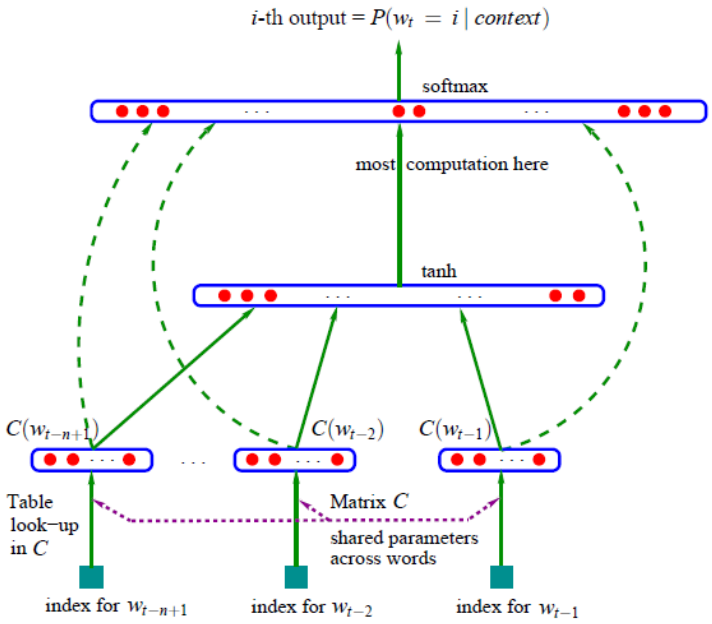


Figure 2.18: A classic neural architecture for word embedding. The function is defined as $f(w_t, ..., w_{t-n+1}) = g(i, C(w_{t-n+1}), ..., C(w_{t-1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector. (Image from Bengio et al., 2003).

Starting from this basic neural model, Mikolov et al. proposed the Word2Vec algorithm, which is tailored around two different architectures, namely Continuous Bag-Of-Word (CBOW) and skip-gram.

CONTINUOUS BAG-OF-WORDS The training objective of this architecture is to predict a given word w_t from a context. The main idea behind the model is to use information from the past and future contexts of given word. Thus, the network take as input both the n words before and after the target word w_t and fine-tune its parameters θ in order to output the right prediction (see Figure 2.19). Therefore the objective function that the network maximizes is defined as follows

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{t-n}, ..., w_{t-1}, w_{t+1}, ..., w_{t+n}) \tag{2–40}$$

We can see that this equation simply means that we try to maximize the log-likelihood over the whole dataset of words inside their respective context, both in the past and the future. In order to compute a given context word probability, this architecture use the softmax function as

$$P(w_t \mid w_{t-1}, ..., w_{t-n+1}) = \frac{exp(h^\top v'_{w_t})}{\sum_{w_i \in V} exp(h^\top v'_{w_i})} \tag{2–41}$$

Where v'_{w_t} is the output embedding of word w and h is the output vector of the penultimate layer in the neural language network. To show the quality of their results, the authors perform simple algebraic operations directly on the vector representation of words. For example, they compute vector $X = vector(\text{"biggest"}) - vector(\text{"big"}) + vector(\text{"small"})$ and they search in the vector space for the word closest to X measured by cosine distance. If this word is the correct answer (here, the word "smallest") the operation is counted as a correct match. By doing so on several examples, they obtain a score that reflects the efficiency of the embedding. For this architecture, with a context size of 10 words and an embedding space representation of 300 dimensions, the accuracy reached is 36.1% on 10000 triplets.

SKIP-GRAM While the CBOW model uses the whole context around a given word to predict it, the skip-gram model performs the opposite task. Hence, the model relies on a single given word input, and tries to predict its whole context words, both in the past and future (see Figure 2.19). Therefore, the objective of the skip-gram is to maximize the probability of a complete context (represented by the n surrounding words to the left and to the right), given that we observe a particular target word w_t. Consequently, the objective to maximize is given by the log-likelihood over the entire dataset of T words

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \leqslant j \leqslant n, \neq 0} \log p(w_{t+j} \mid w_t) \tag{2–42}$$

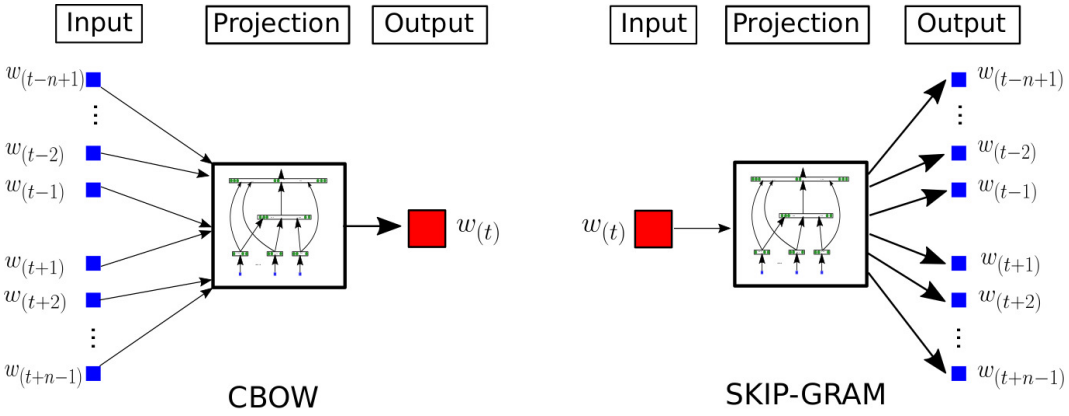


Figure 2.19: Continuous bag-of-words and Skip-gram architectures for Word2vec (*Mikolov et al.*, 2013). The model tent to predict a given word w_t from a context composed by n words before and after the target or vice versa.

In the skip-gram model, instead of calculating the probability of the target word w_t given its previous words, the model computes the probability of a context word w_{t+j} given w_t. Moreover, as the skip-gram model does not use an intermediate layer, in this case, h simply becomes the word embedding v_{w_t} of the input word w_t, which lead to the following equation

$$p(w_{t+j} \mid w_t) = \frac{exp(v_{w_t}^\top v'_{w_{t+j}})}{\sum_{w_i \in V} exp(v_{w_t}^\top v'_{w_i})} \tag{2–43}$$

Even though the task might seem extremely hard to learn, this model was shown to strongly outperform CBOW. Indeed, on the same dataset and same parameters, the accuracy for the superlative task reaches 53.3%.

2.4.2.2 *GloVe*

The main idea behind GloVe is that the ratio between the co-occurrence probabilities of a word given two other words might contain significantly more information than the single cooccurrence probabilities separately (this concept is better detailed in Table 2–1). Hence, it is this ratio that the network will try to encode as a vector representation. Therefore, the input is no longer a stream of words defined by a sliding window but rather a complete word-context matrix of co-occurrences.

To train the model, the authors propose a weighted least square objective J that directly aims to minimize the difference between the dot product of the embedding representation of two words and the logarithm of their number of co-occurrences

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^\top \hat{w}_j + b_i + \hat{b}_j - \log X_{ij})^2 \tag{2–44}$$

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Table 2–1: Co-occurrence probabilities for target words ice and steam with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam. (Table and text from *GloVe: Global Vectors for Word Representation* Pennington, Socher, and Manning, 2014)

where w_i and $\hat{w}_i$ are the embedding vectors of word i and j respectively, b_i and $\hat{b}_j$ are the biases of word i and j respectively, X_{ij} is the number of times word i occurs in the context of word j, and f is a weighting function that allows to assign a relative importance to the co-occurrences given their frequency.

An exact quantitative comparison of GloVe and word2vec is difficult to produce because of the existence of many parameters that have a strong effect on performance (vector length, context window size, corpus, vocabulary size, word frequency cut-off). This question stirred up heated debate in the machine learning community. Despite this, GloVe consistently outperformed Word2vec by achieving better results with even faster training.

2.4.3 *Space representation*

A crucial question still remains unsettled regarding embedding spaces. Indeed, we have just seen that the primary goal of latent space learning algorithms is to consistently reduce the dimensionality of the input space while ensuring a meaningful organization of the samples in the output space. This is why the choice of the latent space dimensionality is a carefully considered trade-off between providing the smallest possible output space and keeping enough dimensions to express all the complexity of the input data. In the literature, these values vary from 30 to 512 dimensions depending on various types of parameters. Since we can not display more than 3 dimensions, it is therefore impossible to really visualize this kind of space as it stands. However, this would seem very convenient in purposes of appraising the overall structure of the space and highlighting relevant geometrical patterns between embedded samples.

That is why we have to rely on algorithms which project high dimensional spaces in displayable ones (e.g. 2D or 3D) while keeping their geometric properties. In the following we present two possible solutions, namely the Principal Component Analysis (PCA) and the t-distributed Stochastic Neighbor Embedding (t-SNE).

2.4.3.1 *PCA*

The PCA is largely the most widely used dimensionality-reduction method of the literature. Based on the work of Karl Pearson (Pearson, 1901), it has been developed and popularized by the statistician Harold Hotelling (Hotelling, 1933). Since then, this algorithm has been extended and adapted to a wide variety of scientific fields taking different names across its different applications. Although existing many ways to describe and process the PCA, its basic concepts can be mathematically defined as follow.

Let is consider a matrix $\mathbf{X} = (x_i^j)_{n\times p}$ of n observations described on p variables where $x_i \in \mathbb{R}^n$ define the i^{th} observation and $x^j \in \mathbb{R}^p$ define the j^{th} variable. First, we standardize the range of the different variables forcing them to have a mean equal to 0 and a variance equal to 1. Thus, we obtain the standardized matrix $\mathbf{Z} = (z_i^j)_{n\times p}$ where

$$z_i^j = \frac{x_i^j - mean(x^j)}{\sqrt{var(x^j)}} \qquad mean(z^j) = 0 \qquad var(z^j) = 1$$

Note that this step is optional yet is used in most cases. Indeed, since the PCA is highly correlated with the variances of the variables, a variable with a larger measurement scale (i. e., a greater variance) will take on too much importance in the analysis compared to the others and thus potentially bias the result.

Then, we aim to find the correlations between variables, or in broader terms, how much the different variables are varying with respect to each other. To do so, we compute the $p \times p$ covariance matrix $\mathbf{K}_{ZZ}$ defined as

$$\mathbf{K}_{ZZ} = E[\mathbf{Z}\mathbf{Z}^\top] - E[\mathbf{Z}]E[\mathbf{Z}]^\top \tag{2–45}$$

We can intuitively understand that highly correlated variables are carrying redundant information making them good potential candidates to be removed or combined into a single variable. In order to precisely quantify the relevance of each original variable in the description of the input data, we compute the *Eigendecomposition* of $\mathbf{K}_{ZZ}$ defined by

$$\mathbf{K}_{ZZ} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{2–46}$$

where $\mathbf{Q} = (q_i^j)_{p\times p}$ is the matrix composed by the *eigenvectors* q_i of $\mathbf{K_{ZZ}}$ and $\mathbf{\Lambda}$ is the diagonal matrix whose elements are the corresponding *eigenvalues* $\Lambda_{ii} = \lambda_i$. The properties of such a matrix ensure that each q_i defines the direction of the axes that maximizes the variance across the corresponding variable, and the attached λ_i indicates the amount of variance it carries. Therefore, by ranking eigenvectors in descending order according to their eigenvalues, we obtain the principal components in order of significance. Finally, we construct the *feature matrix* $\mathbf{W} \in \mathbb{R}^{p\times k}$ by selecting the k main principal components (with $k = 2$ or $k = 3$ in the case of a reduction to 2D or 3D) and project the original data in the space defined by this new basis to obtain the reduced representation of the input data.

$$\mathbf{X}_{PCA} = \mathbf{W}^\top \mathbf{Z}^\top \tag{2–47}$$

In that way, it become possible to display spaces with more than three dimensions. However, we must not forget that dimensionality reduction processes inevitably lead to a loss of information, even if we tend to minimize it. Moreover, as linear combinations of the initial variables, the principal components are not carrying real meanings anymore nor physical interpretations but are only partially reflecting the geometrical relationships between observations. Finally, this linear nature could lead to poor visualizations especially when dealing with non-linear manifold structures.

2.4.3.2 *t-SNE*

One of the most efficient alternative to the PCA, called t-SNE, has been proposed in Maaten and Hinton, 2008b. This unsupervised learning algorithm is an improved version of the original Stochastic Neighbor Embedding (Hinton and Roweis, 2002) and has proven to be very effective in the visualization of non-linear high-dimensional data distributions on displayable spaces.

Starting again from a matrix $\mathbf{X}_{n\times p}$ composed by n p-dimensional observations, we first aim to measure the similarities between elements in the original high dimensional space. To do so, we rely on the density of all points x_j under a Gaussian distribution centered around a point x_i by computing the conditional probability $p(j\,|\,i)$ as

$$p(j\,|\,i) = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \qquad with\ \forall i \quad \begin{cases} i \neq j \\ p(i\,|\,i) = 0 \\ \sum_j p(j\,|\,i) = 1 \end{cases} \tag{2–48}$$

Through this, we can calculate the joint probabilities $p(i,j)$ which reflect the similarities between the points x_i and x_j.

$$p(i,j) = \frac{p(j\,|\,i) + p(i\,|\,j)}{2n} \qquad with \quad \begin{cases} p(j,i) = p(i,j) \\ p(i,i) = 0 \\ \sum_i \sum_j p(i,j) = 1 \end{cases} \tag{2–49}$$

Indeed, if the probability that two distinct points x_i and x_j are under the same local Gaussian distribution are high, this means that these two points are highly similar. We can adjust the locality of this analysis (i. e.take into account a more or less wide neighborhood) by modifying the bandwidth of the Gaussian kernels expressed by σ_i. The authors have shown that this parameter, called *the perplexity*, should stay in the range between 5 to 50 to ensure a normal behavior of the algorithm.

Then, we define another set of joint probability $q(i,j)$ which reflect the similarities of the elements in the d-dimensional target space $\mathbf{Y}_{n\times d}$ (with $d = 2$ or $d = 3$ in the case of space displays). We proceed as for p except that this time we use the heavy-tailed Students t-distribution with one degree of freedom (or Cauchy distribution) instead of a Gaussian distribution. Indeed, its properties allow a better modeling of dissimilar elements thus located far from each other in the output space. Therefore, for $i \neq j$, $q(i,j)$ is defined as

$$q(i,j) = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l\neq k} (1 + \|y_k - y_l\|^2)^{-1}} \tag{2–50}$$

Since the goal for the low-dimensional space $\mathbf{Y}$ is to mirror the local similarities of $\mathbf{X}$, the set of probabilities $q(i,j)$ must be as close as possible to $p(i,j)$. Hence, we rely on the KullBack-Leibler Divergence ($\mathcal{D}_{KL}$) to measure the difference between both distributions as

$$\mathcal{D}_{KL}[P\|Q] = \sum_{i\neq j} p(i,j) \log \frac{p(i,j)}{q(i,j)} \tag{2–51}$$

Finally, by minimizing the $\mathcal{D}_{KL}$ with respect to $\mathbf{Y}$ through gradient descent, we learn a map which reflects the similarity between the elements in the high-dimensional space.

2.5 SYMBOLIC MUSICAL SPACES

We have seen that the representation of symbolic music has taken a central role in the evolution of musical knowledge and creativity, thus becoming one of the key

challenges of computer music. Moreover, the success of embeddings in the NLP field has raised the perspective of representing musical objects in a relatively low-dimensional space organized according to fundamental concepts of music theory. Therefore, several attempts have been made during the last decade to adapt the embedding approach to musical data. In this section, we present the main proposals that have made progress towards this objective. We can divide these works into two categories, namely the *Prediction-based* which consists of mimicking the techniques of the NLP field, and the *VAE-based* which rely on the Variational Inference paradigm.

2.5.1 *Prediction-based*

One of the first direct adaptation of *Word2Vec* has been proposed in Huang, Duvenaud, and Gajos, 2016. In this work, the musical units that have been embedded are simple chords described through their main attributes such as their root, their type (major, minor, diminished), inversion, extensions and alterations. The resulting strings are used as input to train the skip-gram version of *Word2Vec* (see Section 2.4.2.1) on two different datasets composed respectively by the chorales written by J-S Bach and 200 rock song from *the Rolling Stone Top 500 Hits*. A simple PCA projection of the 10-dimensional learned embedding with two principal components has highlighted interesting topological properties, with Minor and Major chords following the circle of fifth. Moreover, it seems that the distances between chords in the space are partially reflecting musical similarities in term of functional harmony. Thus, despite the simplistic nature of this approach which makes the resulting space unusable in realistic contexts, the authors have confirmed the relevance and the potential of the embeddings framework for symbolic music.

In Madjiheurem, Qu, and Walder, 2016, another way to obtain a musical version of *Word2Vec* has been explored. In this approach, the embedded unit is still the chord although this time there is no limitation as to the number or type of chords to be accounted for. Any different stacking of notes occurring throughout the dataset is considered a chord and is represented as a *Piano-roll* matrix. Three models inspired from the NLP field has been trained including again the skip-gram version of *Word2Vec*. The second one is based on the Neural Autoregressive Distribution Estimator (NADE) which are very efficient for modeling the distribution of high-dimensional vectors of discrete variable (Larochelle and Murray, 2011). Finally, the last one is an architecture called *Sequence-to-sequence* (or *Seq2Seq*) (Sutskever, Vinyals, and Le, 2014). These particular auto-encoders are composed of RNNs for both, the encoder and the decoder, and aim to encode an entire input sequence in a latent space from which another sequence will be inferred. In

this paper, these three models have been trained through the prediction task on four different datasets. Based on the log-likelihood evaluation, the *Seq2Seq* model shows superior results. However, the authors did not go further in the evaluation of their embedding spaces.

In Boulanger-Lewandowski, Bengio, and Vincent, 2012, the predictive models that have been proposed for learning musical embeddings are based on Restricted Boltzmann Machine (RBM) (Smolensky, 1986). These specific stochastic neural networks aim to learn the probability distribution over the set of inputs in order to project it in a lower dimensional space. By combining it with RNNs, the authors have designed the most successful model in the literature used for predicting musical events. As in the previous approach, the input data are *Piano-roll* vectors describing the note activations across a discretized time axis. To allow the encoding of all the events included in the whole set of scores, this time reference quantum has been fixed to the eight note. However, as the average duration of a note is much longer, this leads to a representation with a huge amount of repetitions when dealing with entire musical sequences. Under these conditions, a network trained trough the prediction task will simply learn to repeat the last event as it create a very strong local minimum in the loss function. To avoid this, the authors have relied on the *event-level* prediction by considering only the frames where at least one pitch differs from the previous one. As this study have been hardly focusing on the prediction scores, the structure of the latent space have not been explored.

Bretan et al., 2017 have introduced the use of CNN for learning musical embedding space in the symbolic domain. They have also brought a new perspective to this field of research by considering four consecutive beats as the basic unit to be embedded rather than a single chord. The model they used stems from a class of AE called *denoising Auto-encoder* whose aims is to learn to *denoise* a corrupted version of the input. In this paper, the decoder is the mirror network of the encoder which is composed by four convolutional layers followed by three fully-connected layers. As in the previous approach, the authors have relied on the *Piano-roll* representation limited to the note onset frames to avoid repetitions. Moreover, a regularization of the embedding space has been applied by using the task of composer classification as a means for the training. The quality of the learned space has been assessed by using the embedding vectors as input representations for training neural networks on the forward prediction and composer classification tasks. The whole system have been performing well on the second one, however, as no other topological properties of the space have been highlighted, this seems to result mostly from the regularization technique rather than the embedding.

2.5.2 *VAE-based*

More recently, the VAE (see Section 2.3.2.5) have provided an elegant approach for learning embedding spaces. Thanks to the constraint that the code z is a random variable distributed according to a prior $p(z)$, the resulting latent space is forced to be smoothly organized without any region of the space which do not map with realistic data. Hence, in addition to the low-dimensional representation of the inputs, this approach offers the possibility to generate new realistic data directly from the space by sampling latent codes from $p(z)$. These paradigm has been used in a wide variety of domains leading to powerful applications such as natural image and text modelling (Oord et al., 2016; Bowman et al., 2015). In the field of symbolic music, one model in particular, called *MusicVAE*, has demonstrated the strength of this approach (Roberts et al., 2018).

2.5.2.1 *MusicVAE*

The goal of this model is to accurately project short monophonic melodies (from 2 to 16 bars) in a reduced latent space. To do so, MIDI files used as training inputs are sliced in few bars and represented using a simpler version of the MIDI-like representation (see Section 2.2).

The model itself is defined as a recurrent VAE. The encoder is a two-layers bidirectional LSTM network that produces a sequence of hidden states $h = \{h_1, h_2, ..., h_T\}$ from an input sequence $x = \{x_1, x_2, ..., x_T\}$. The final encoding z is then set as a function of the last hidden state h_T. For the decoder, the authors have proposed a novel hierarchical recurrent neural network composed by two LSTM network. The first one, called the conductor RNN, segments the output target into U non-overlapping sub-sequences and produces an embedding vector $c = \{c_1, c_2, ..., c_U\}$ for each time step. Finally, the last LSTM network auto-regressively produces a sequence of distributions over output tokens for each sub-sequence via a softmax output layer. The architecture is depicted in Figure 2.20. This hierarchical system allows to address the issue of long-term structure modelling that occurs when using VAE. Indeed, the authors have shown that the reconstruction accuracy are very poor when using a simple RNN because of the vanishing influence of the latent state as the output sequence is generated.

Several tests have been set up in order to validate the quality of their proposal. First, the authors emphasized the logical structure of the learned space by showing that the generation of an interpolated sequence between two points is producing a smooth musical evolution. In addition, they have explored the *attribute vector arithmetic* technique which consists in altering the musical attribute of a sequence

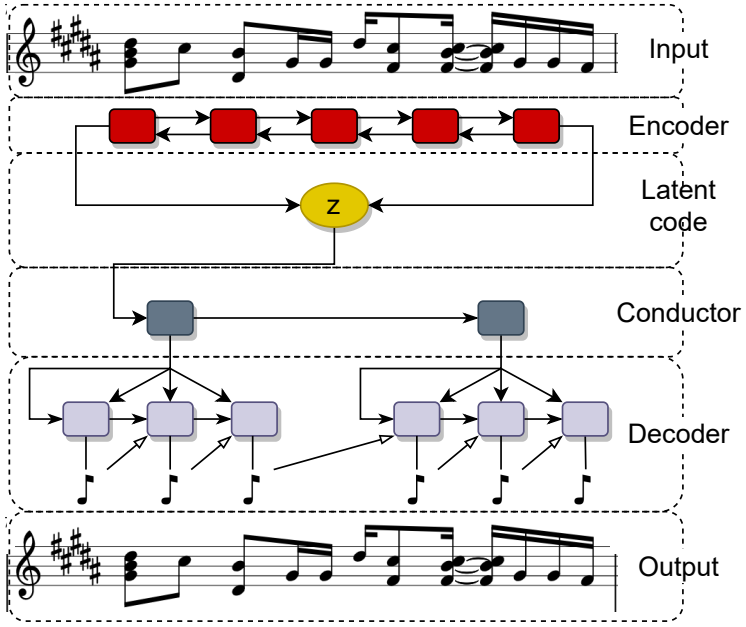


Figure 2.20: *MusicVAE* architecture with a hierarchical decoder. The conductor RNN provides sub-sequences from which the final output is recursively decoded by the decoder RNN (Roberts et al., 2018).

through its latent code. To do so, they have defined five attributes for each samples, namely C diatonic membership, note density, average interval, and 16th and 8th note syncopation. By averaging the latent codes for all datapoints which possess a given attribute, they obtained a vector which can be used to make targeted changes to data examples. In their contexts, the intended changes were mostly well executed despite the appearance of different side effects.

Thus, *MusicVAE* has shown very interesting results and the latent space obtained from it seems to be usable for creative applications. Nevertheless, it only takes into account monophonic melodies which drastically limit its scope of use as polyphony is a crucial notion in music.

2.6 CONCLUSION

In this chapter, we have provided the details necessary to understand the context in which we conducted our research and the trends that have been taken. We have first laid the foundations for the symbolic representation of music and the challenges it raises for computer science. Then, we have explained in detail the basic mechanisms of ML and the more advanced models we ran with. Finally, we have

introduce what has been the core of our research, the *embedding spaces*.

We have seen in the first section that the *representation* are playing a major role in the evolution of the music. Indeed, the improvements brought to musical notation through the ages have led to the development of increasingly complex compositional techniques with the appearance of new rhythmic and harmonic patterns. At the end of the last century, with the rise of the computer science, the digital era further broadened new horizons for musical creation that had previously been unthinkable. However, we have seen through the description of the three major representations of the literature that representing music in an efficient computer-oriented way is a difficult task. In spite of its universal aspect and its simplicity of use, the *piano-roll* still possesses some undesirable properties in computer science such as its highly sparse and repetitive nature. The *MIDI-like* representation does not suffer from these limitations but nevertheless seems difficult to manage in the context of polyphonic music because of the large size of the vocabularies used to describe the evolution of the notes and the disorder in which they occur. The *Note-Tuple* representation addresses both of this issues by relying on small vocabularies ordered in tuples which, however, results to the necessity to employ six different vocabularies to describe a single note. Finally, we have put forward the merits of musical spaces by introducing three different ones built from mathematical backgrounds that allowed us not only to develop our musical knowledge but also to compose original music.

The second section has been dedicated to the ML framework. We have seen that a neural network can be seen as a non-linear transformation of an input space into a new output space. This transformation is driven by a loss function that the network tries to minimize by optimizing its various parameters through the *gradient descent* algorithm. For this reason, this paradigm seems to be a good solution to reach our goal of finding a meaningful representation of symbolic music. In addition, we have presented some more advanced ML mechanisms that make it an even more attractive alternative. We have seen that thanks to the convolution operation or to well designed temporal loops, a Neural Newtork (NN) could capture fine spatial and temporal features of an input. Moreover, we have presented the class of *auto-encoder* networks which aims to compress information in a relatively low-dimensional space. this process can be enhanced through the use of the *attention mechanism* which provide a hierarchy regarding the relevance of each element in the input sequence, or through the regularisation of the latent space by imposing the output to be close to a prior distribution.

As seen in the third section, the idea of using ML algorithms to learn a meaningful representation of a given type of data has appeared in the NLP field. The structure of these famous *word embeddings* reflects the semantic relationships between words and thus provides precious information on them which can be used for handling complex task. We have shown by describing the two most striking models in the literature (*Word2Vec* and *GloVe*) that the training of these spaces was done through the prediction of events according to a context. By doing so, the model learns the probability that a given word occurs in a context which is a good indicator of its overall meaning.

Following the success of this approach, multiple attempts to built musical embeddings has been made. The main papers we have presented have all made a valuable contribution, and even if the spaces obtained are not as efficient as the word embeddings, increasingly encouraging results have been obtained. In the meantime, a novel trend relying on the so-called VAE has make its appearance in the stream of embeddings research. Based on this probabilistic class of architecture, we have presented an outstanding model called *MusicVAE* which provides powerful latent spaces for symbolic music but only for monophonic data.

3

PREDICTION-BASED FRAMEWORK

3.1 INTRODUCTION

The aim of this thesis is to design algorithms for learning musical embedding spaces in the symbolic domain. In other words, we want to capture information about the semantic relationships between musical items in an unsupervised manner, and to encode it in a low-dimensional space. In this chapter, we propose a method to tackle this objective, by taking inspiration from the NLP field, since we rely on a *prediction task*.

We propose in Section 3.2 a new model which takes its roots from the critical differences that exist between textual and musical data. First, we explain the reasons behind our different choices before presenting the architecture in detail. Then, we introduce a new AM derived from *Multi-head Attention Mechanisms* (Vaswani et al., 2017), aiming to enhance the overall performances of our system. Finally, we describe the training procedure as well as the datasets on which we have tested it.

We define our evaluation method in Section 3.3, where the results of our proposal are presented in two categories. The first gathers the different prediction results obtained according to the model used and the dimensionality of the embedding, while the second presents a topological study of these spaces. Finally, we conclude in Section 3.4 by summarizing and discussing our different results.

3.2 CNN-LSTM MODEL

Here, we introduce a novel model, which belongs to the general class of *auto-encoders* (see Section 2.3.2.4). These two-parts networks are specifically designed to compress input data with the least possible loss of information. Thus, in theory, redundancies contained in the input representation are removed and the pertinent information is encoded in a latent vector.

3.2.1 *Motivations*

As we have seen in Section 2.4 with word embeddings, a direct approach to optimize a model able to disentangle the underlying semantic of a set of data is

to train it on a *prediction task*. Indeed, the probability that an event occurs in a given context is highly tied to its meaning. This implies that two elements with a high probability of occurrence within the same context will be close semantically. Furthermore, we can also intuitively understand that prediction tasks are greatly simplified if the input data space is logically ordered. For these reasons, we choose to rely on this paradigm in order to address our objectives.

However, as seen in Section 2.5, a direct application of the word embedding methods leads to significantly weaker results on symbolic music. Our assumption to explain these results is that, in spite of the apparent similarity between musical and textual languages, some fundamental characteristics of music are not well handled by word embedding methods. This considerably limits the amount of relevant semantic information extracted by the network. Indeed, a text is composed by a very *wide variety* of symbols (words) that appear *sparsely* across the data, while musical scores are defined by *few* symbols that re-occur *frequently* along the score. Moreover, the crucial notion of *transposition* in music does not exist in text data, while we would expect any musical embedding space to deal with it adequately. Another musical aspect that word embedding algorithms are not able to handle correctly is to separate rhythmic information. In these models, there are no components that aim to capture information about time dependencies.

Hence, given these observations, we tried to build a model that could address all of these shortcomings through two main modules. The first one is a CNN aiming to capture the different structures and shapes formed by the note activations in the piano-roll frames. By applying convolution along the pitch axis, we expect the learned feature maps to reflect the core properties of musical chords, namely the number of notes and the intervals between each note. Moreover, by choosing a kernel size equal to 12 – corresponding to the number of notes in an octave in Western music – there will be some common kernel activations between two identical chords at different root notes, thus leading to common features in the latent code (see Figure 3.1). The second module is an LSTM, whose role is to target fine temporal relationships in a sequence of consecutive piano-roll frames since there is much more subtlety in the rhythmical ordering of events in a musical sequence than in a textual one.

In addition, one very important thing to consider when dealing with sequential data is the relative importance of each element in the sequence. For example, in order to understand the general meaning of a sentence, we devote more importance to verbs and nouns than to articles and other linking words. Similarly, adjectives are sometimes predominant in conveying meaning and other times are almost

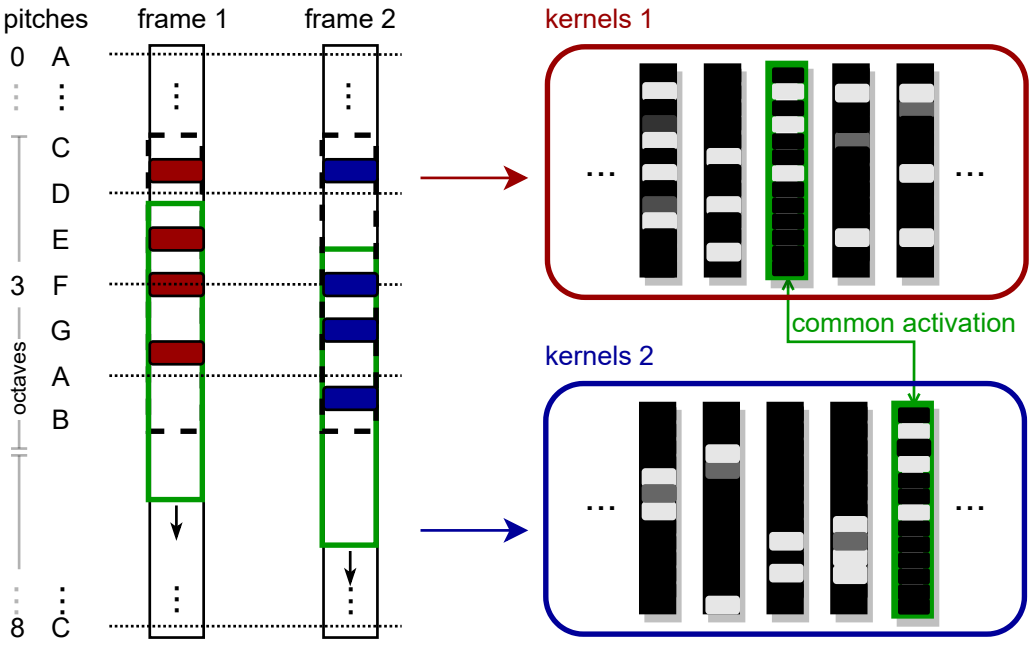


Figure 3.1: Octave convolution applied on piano-roll frames.

purely decorative. As we have seen in Section 2.3.2.6, the so-called *attention mechanism* was specifically designed to leverage on this observation. In our context, this module could be very valuable since these hierarchical relationships are strongly present in music. We can even go further by noting that, in the context of polyphonic music, this classification according to the importance of each element can also be carried out *vertically* within the same piano-roll frame. Indeed, a chord can be embellished with different notes that will not fundamentally change its nature and which will, therefore, have to have a less prominent role in the encoding. For these reasons, we propose an AM specifically designed to weigh the elements of a musical sequence *temporally* (horizontally) and *harmonically* (vertically) that we called Hierarchical Attention Modulation (HAM).

3.2.2 *Architecture*

The proposed model is based on an encoding network architecture, as depicted in Figure 3.2. To learn a structured embedding space reflecting musical properties, the model encodes each event of a sequence in an embedding vector and tries to predict the next event based on this representation. Then, this prediction is decoded and compared with the ground truth. In order to ensure that the model learns an adequate embedding rather than solely optimizing the prediction task, almost all of the network capacity lies in the encoder. We ensure this by relying on

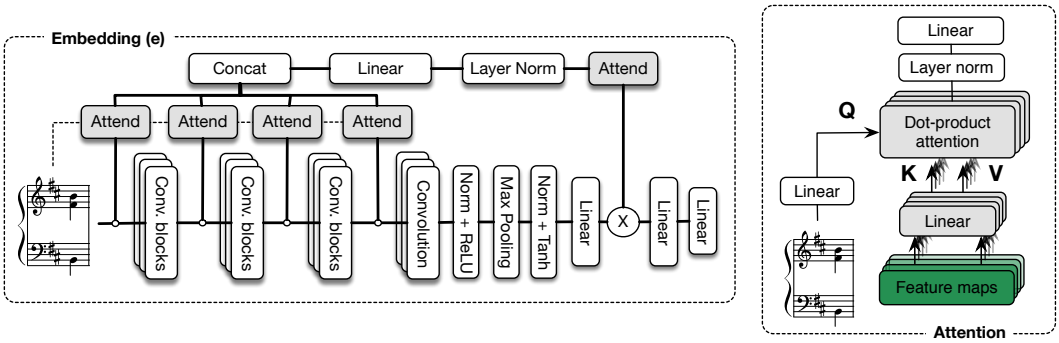


Figure 3.2: Our proposal is composed of two major parts. (*Left*) A convolutional network is augmented with a separate attention mechanism at each convolutional layer. This allows to attend the most salient information at each level of processing. (*Right*) The input acts as a query to the attention module, which works on each of the feature maps separately.

very low-capacity prediction and decoder models (see Figure 3.3).

The embedding architecture is based on a CNN. In addition, we introduce a novel attention mechanism called HAM. The main idea is to attend each convolutional layer of processing separately, in order to obtain a multi-scaled hierarchical view over the different levels of abstraction. The attention module itself operates on a kernel-based operation instead of the spatial attention, by processing each feature map independently. This proposed attention module is developed in detail in the next section. Furthermore, the input is used as a query across all attention modules rather than the feature maps themselves. Information from the different hierarchies of attention are then concatenated and mixed before being attended again with a simpler dot-product attention. Finally, this hierarchical attention information is used to modulate the fully-connected transform of the encoder.

The final model is composed of 5 convolutional layers with 300 channels each, except the last one with 200 channels. As an octave is composed of 12 notes in western music, we impose transposition-invariance activations by relying on kernels of size 12. We rely on the ReLU as non-linearity and apply batch normalization and dropout (with a factor of $p = 0.4$) between each layer. At the end of the convolution processing, we perform a max-pooling to reduce the dimensionality by a factor of 2. Finally, 3 fully-connected layers of 1500, 500 and d_e units combine the activations of the convolutions filters modulated by the hierarchical attention modules, where d_e is the dimensionality of the embedding. This leads to output a final embedding vector of d_e dimensions. We conduct a benchmark to assess the impact of d_e on the overall performances of the system and to choose the best

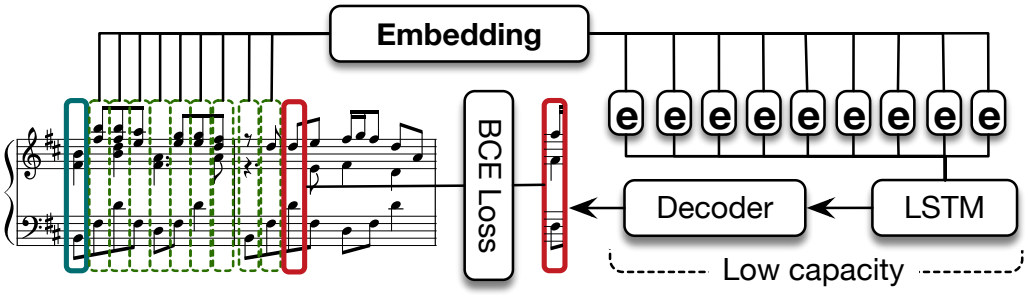


Figure 3.3: The second part of our model is a very-low capacity LSTM network to ensure that the embedding learns a structured space. This layer predicts the event following a sequence of embedded events. The decoder mirrors the encoder to output the predicted piano-roll frame.

alternative. The results are displayed in the following section.

The second part of the model is a simple 1-layer LSTM network with only 500 units to predict the events in the embedding space. It is fed with encoded sequences of 12 elements leading to an input matrix of size $12 \times d_e$. Finally, the predictions are decoded by mirroring the operations performed by the encoder in reverse. This produces a piano-roll frame, which is evaluated by using a Binary Cross Entropy (BCE) criterion, defined as

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{3–52}$$

where N is the length of the piano-roll frames, $\hat{y}_i$ is the i-th value of the predicted frame and y_i is the corresponding target value.

3.2.3 *Hierarchical attention modulation*

Here, we propose to rely on a hierarchical attention mechanism inspired by multi-head attention, but that differs in several key aspects. First, the hierarchical aspect comes from the fact that the attention is applied at each layer separately to select the most relevant information. However, this information does not feed the following layers but is rather mixed and attended again separately. The resulting output is used to modulate the last fully-connected layers of the network. Finally, the attention module itself targets entire feature maps rather than spatial locations and can be mathematically described as follows. First, each feature map is processed through a linear transform separately to reduce its dimensionality to d_t. This produces a set of keys K and values V. The query Q is defined by the output of

the preceding convolutional layer. The network computes the scaled dot-product attention of each reduced feature map as

$$Attention(Q, K, V) = Softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{3–53}$$

Finally, this set of attention vectors is normalized and mixed across kernels to obtain a summary of the prominent information at each given layer. The whole process is depicted in Figure 3.2

3.2.4 *Data and training*

In order to evaluate our proposal, we train it on the four reference symbolic prediction datasets that have been the most widely used in the literature.

- **JSB Chorales** is a corpus of 382 chorales by Johann Sebastian Bach, which is split into the train, test and validation sets proposed by Allan & William (Allan and Williams, 2005).
- **Nottingham** is a collection of 1200 British and American folk tunes[1].
- **Piano-midi.de** is a collection of classical music played on piano, following the split by Poliner & Ellis (Poliner and Ellis, 2006)[2].
- **MuseData** is a dataset developed by the CCARH that includes 880 orchestral pieces of famous composers[3].

For all datasets, we extract the piano-roll matrices from the MIDI files by quantifying the time scale at 16 frames per beat and set played note values to 1 and 0 otherwise. Hence, the resulting matrices are composed of a large majority of repeated frames. As shown by Crestel and Esling, 2016, in this *frame-based* context, a dull model which simply repeats the last musical frame outperforms all previously proposed models (with an accuracy of up to 85% on the JSB dataset). A solution to avoid this problem could be to predict an entire sequence rather than a single note. However, all previous models are evaluated on single-frame prediction tasks and learning on entire sequences could render the learning very unstable and deprive the resulting embedding of its metric properties. Therefore, we constrain the prediction by transforming piano-roll matrices into an *event-level* representation (keeping a single frame per new event). We keep the reference dataset splits if available, otherwise we follow a 80% train and 20% test split.

1 https://ifdo.ca/ seymour/nottingham/nottingham.html
2 http://www.piano-midi.de/
3 http://www.musedata.org/

Based on the sequence of events, we train our network to predict the next musical event. As the great majority of the capacity is in the encoder, the goal is that the model improves the structure of the embedding to improve its prediction score. We feed the network with mini-batches of size 32, each element being a sequence of 12 events (the 128-dimensional frames). All the events are encoded in the d_e-dimensional space and the LSTM network tries to predict the 13^{th} event of the sequence by relying on the previous 12 embedded vectors. Then, the predictions are decoded by processing the encoding reversely and the 128-dimensional resulting vectors are compared to the real target through the BCE loss. Regarding optimization, we use the ADAM learning algorithm (Kingma and Ba, 2014), with an initial learning rate of 10^{-5} and use an optimization scheduler that halves the learning rate after 25 epochs without improvements.

3.3 METHOD EVALUATION

In order to assess the quality of our proposal, we perform two separate types of analyses. The first one consists in comparing the prediction results of our model with those of the state of the art. In addition, we also evaluate the scores obtained by performing gradual ablations of our network in an attempt to determine the contribution of each module. Finally, we study the impact of the embedding dimensionality on these prediction scores. On the other hand, the second evaluation aims to analyse the structure of the embedding in a visual manner. To do so, we display several PCA projection of selected elements according to their musical properties and we observe the geometrical patterns which emerge from these graphs.

3.3.1 *Prediction results*

In order to evaluate the success of various models, the frame-level accuracy of the prediction is computed for each piano-roll frame in the validation set. This measure was specifically designed for evaluating the prediction of a sparse binary vector. Indeed, a purely binary measure corresponding to either a perfect match between the prediction and the ground truth, or an error in any other case, might not reflect the true success of the underlying algorithm. For that reason, the frame-level accuracy measure is usually used to alleviate the problem.

To obtain this measure, we first compute three variables depending on the prediction vector and the ground truth one. The true positives TP is the number of active notes that the model predicts correctly, the false positives FP is the number of 1 which should be 0, and the false negatives FN is the opposite (0 that should be 1). Note that we do not take into account the true negatives values due to the

sparsity of a pitch class vector. Indeed, this property of sparsity leads to a very high number of true negatives (a wide percentage of 0 in both vectors), which might skew the measure by artificially inflating the success of these algorithms. Hence, from these three variables, we can calculate an overall accuracy score for a given predicted vector as follows

$$Acc = \frac{TP}{TP + FP + FN} \tag{3–54}$$

However, we can see that this measure fails to account for rightfully predicted rests (vectors filled with only 0), as even with a perfect prediction, we obtain $TP = 0$. To account for this case, we introduce in this work a new measure of accuracy where a specific term is defined for the rests and the global accuracy score is computed for N predicted vectors with at least one active unit and M vectors of rest as

$$Accuracy = \frac{1}{M + N}\left(\sum_{n=1}^{N}\frac{TP_n}{TP_n + FP_n + FN_n} + \sum_{m=1}^{M}\frac{1}{1 + FP_m}\right) \tag{3–55}$$

This proposed measure of overall performance is now bounded between 0 and 1, where 1 corresponds to perfect prediction.

3.3.1.1 *Architecture analysis*

In order to evaluate the different parts of our proposal, we train several variants of our model by performing an ablation study. First, we evaluate a baseline CNN model (with the same architecture as our proposed encoder), along with a residual and dense version of it. Then, we evaluate different variants of the AM. First, we change the type of attention module used, either with a simple dot-product attention (*AM-dp*) or with the multi-head attention (*AM-mh*) and, lastly, with our proposed kernel attention (HAM). Finally, we replace the modulation aspect of our HAM by a simple addition (*HA+*). In addition to the random baseline model, we compare it to the *RNN-RBM* and *RNN-Nade* models proposed by Boulanger-Lewandowski, Bengio, and Vincent, 2012. The results for the *event-level* prediction are presented in Table 3–2.

First of all, we can notice that the results achieved with the classic CNN module are better than those obtained through its two more sophisticated versions. This result is quite unexpected as the dense and residual CNN have yielded consistent improvements in accuracy across several competitive datasets (see Section 2.3.2.2). Here, we hypothesize that the visual features of the piano-roll frames are relatively straightforward compared to those of images making the reuse of feature across

	JSB Chorales	**Piano-midi.de**	**Nottingham**	**MuseData**
Models	Acc. (%)	Acc. (%)	Acc. (%)	Acc. (%)
RNN-RBM	33.12	28.92	75.40	34.02
RNN-Nade	32.11	20.69	64.95	24.91
Random	4.42	3.35	4.53	3.74
CNN	25.73	22.48	62.31	26.73
Residual	14.85	12.29	53.42	12.30
Dense	15.36	12.74	56.42	16.44
AM-dp	33.61	30.17	64.11	27.17
AM-mh	35.19	32.68	64.25	32.15
HA+	39.07	33.27	76.09	37.84
HAM	**40.25**	**35.28**	**76.25**	**38.15**

Table 3–2: Prediction results measured with the frame level accuracy on the different datasets with an embedding dimensionality $d_e = 30$.

layers less necessary. Thus, the different connections present in the residual modules might hinder the learning of the model instead of improving its performance.

On the other hand, the different attention modules greatly increase the prediction accuracy of the model. Indeed, even with the simple dot-product module (AM-dp), the scores are already significantly better across all the datasets. As expected, the use of the multi-head mechanism also further improves the performance of our proposal to the point of obtaining results already comparable to the best model of the state of the art.

Finally, our complete model (HAM) outperforms all the previous architectures (including the state-of-art ones) on all datasets. We can notice that a stronger modulating signal (by relying on a product instead of a simple sum) on the fully-connected layers with our attention matrices leads to higher scores. This confirms the merits of our approach and demonstrates the benefits of establishing both a temporal and a harmonic hierarchy between the elements of a musical sequence in order to efficiently encode it in a low-dimensional space.

3.3.1.2 *Embedding dimensionality*

One of the fundamental questions when dealing with embedding systems is the dimensionality of the output space. Indeed, since our goal is to compress the information contained in the input data, we want to reduce the number of dimensions of the embedding as much as possible. However, the latent code must contain

enough information to both reconstruct the original input data, while encoding high-level musical theory features. In order to find the best trade-off between these objectives, we compare the prediction scores of our model as a function of the number of embedding dimensions. The results for $d_e \in [10, 30, 50, 100]$ are presented in Figure 3.4.

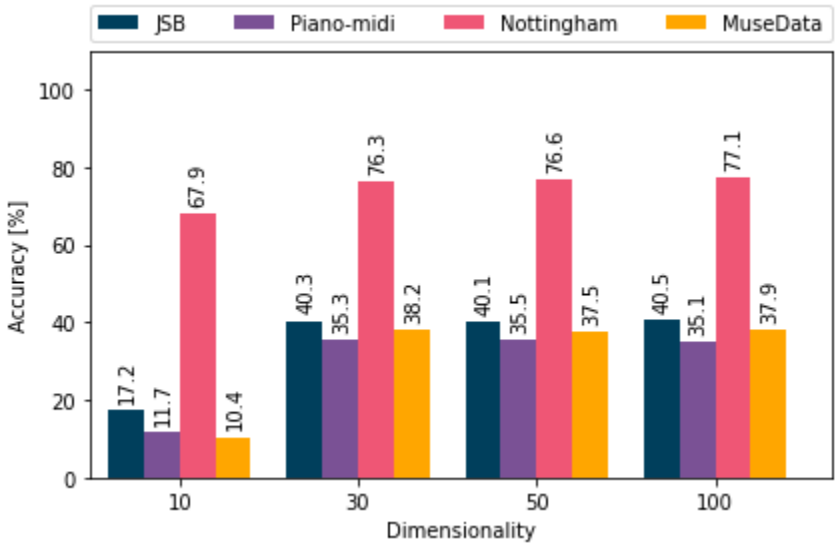


Figure 3.4: Prediction accuracy score for our complete HAM model according to the dimensionality of the embedding space.

As we can see, the best results seems to be attained for $d_e = 30$. In the case of a low-complexity dataset (such as Nottingham) the number of dimensions can logically be further reduced without really compromising the performance of the network. However, in any case, increasing the number of dimensions beyond 30 does not seem to improve the overall performances of our algorithm.

ORTHOGONALIZATION ISSUE By scrutinizing the values of the embedding vectors for each event, we observed that many dimensions were being set to zero (different ones for each event), regardless of the dimensionality of the final space. By doing so, the model tends to *orthogonalize* the distribution of the data in the embedding space during the training. This behavior is quite common in the training of classical AE, where the model focuses only on the reconstruction problem without putting constraints on the space itself. In this regard, the easiest way for the model to correctly discriminate between inputs is to embed them in very different regions of the space. This observation is unfortunate since this behavior does not provide continuous ordering of the elements according to their musical meanings.

3.3.2 *Embedded data visualization*

Here, we provide our second analysis, which is intended to geometrically assess the overall structure of the resulting embedding spaces. To that aim, we compute a PCA on the 30-dimensional space, in order to project all the musical events of the datasets in the corresponding 2-dimensional map (see Section 2.4.3.1). Then, we explore the geometric patterns formed by plotting only musically-related items, as it has been done for word embeddings (see Figure 2.17). We depict several selected examples in Figure 3.5. In these particular examples, we link root notes with their corresponding major 3rd, 4th, 5th, 7th and octave chords with colors depending on their pitch class.

As we can see, it appears that some musical relations have been well captured by the embedding. The majority of resulting segments share some clear common geometrical properties like their length and direction. Nevertheless, few of them do not respect the overall pattern despite the fact that intuitively they seem to be close semantically. This observation appears to be particularly strong for events that occur only very rarely across the whole dataset (such as very high or very low pitches and rather uncommon chords). Our major assumption regarding this issue is that the underlying semantic meaning of a musical event strongly depends on its context. Indeed, a specific chord could have many different "roles" from a musicological point of view depending on where it occurs in the melody or depending on the key signature of the piece.

Moreover, it appears that, in certain cases, the different events are squeezed in the center of the space, thus forming a cluster of very small vectors that are unrelated to the general structure. This is particularly marked on the example of the 4th at the bottom left of Figure 3.5. This phenomenon, which seems to strongly alter the organization of the elements in the space, might be caused by the orthogonalization mechanism discussed previously. Indeed, by setting several coordinates to zero to easily separate elements, the model artificially partitions the space into orthogonal sub-planes whose covariance is zero. Thus, certain sub-planes will "dominate" the PCA (those where the variance of the elements is the highest) and, thereby, bias the principle components that will express the variation of only parts of the data. A solution to alleviate this problem could be to compute different PCA for each sub-planes. However, as this analysis aims to assess the overall geometrical structure of our embedding, it is of limited interest.

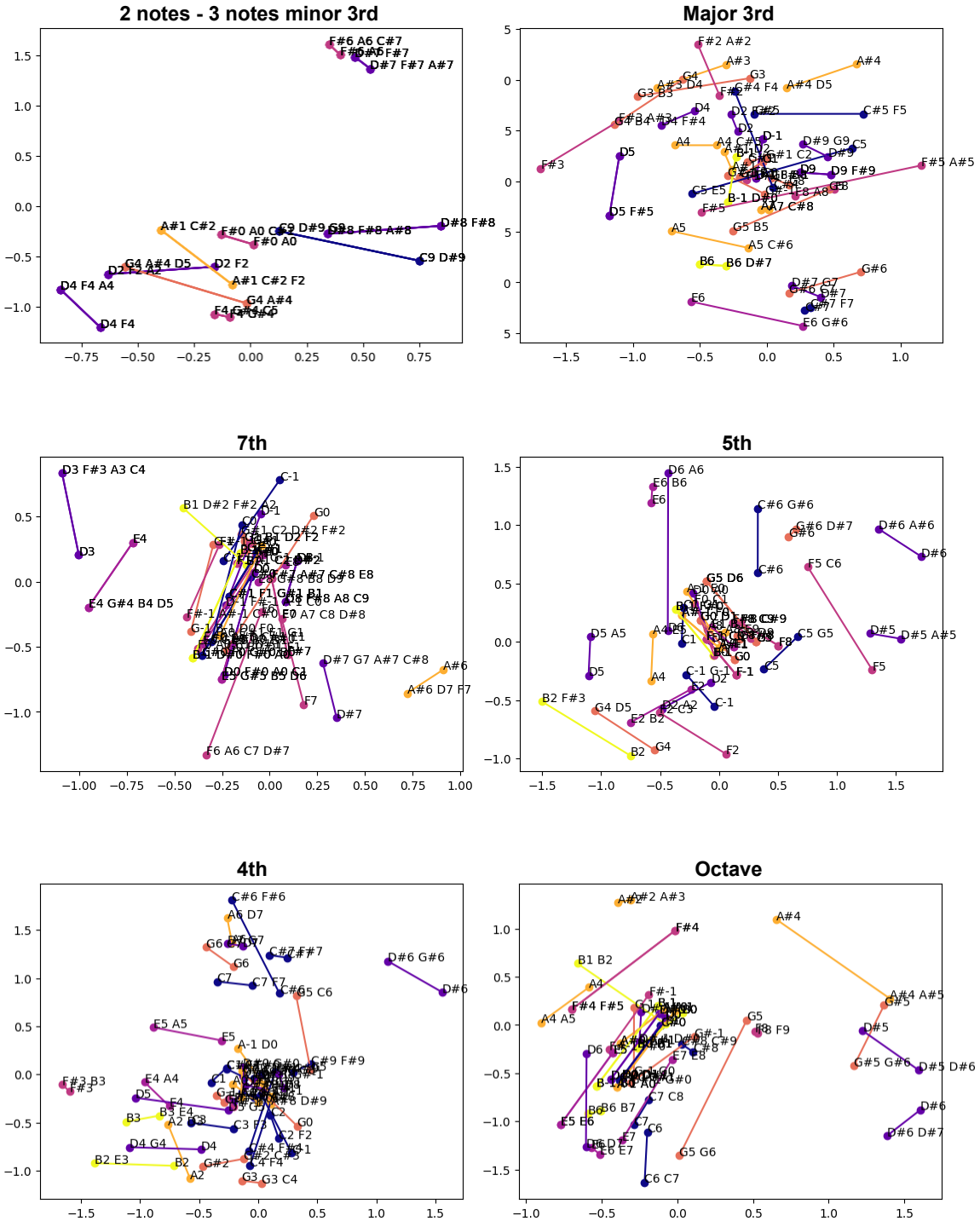


Figure 3.5: Example of relations that emerge from our embedding. Colors depend on the root note. Each note are linked to the corresponding chord.

3.4 CONCLUSION

In this chapter, we have presented a method inspired by the NLP field, which aims to provide musical embedding spaces. To do so, we have proposed a new AE architecture composed by a CNN and an LSTM network both targeting to capture relevant musical features. As the majority of the computational capacity of our model lies in the encoder, by training it on the forward prediction task, we push the model to organize the embedded elements in a logical manner. In addition, we have introduced a novel AM, called HAM, which aims to establish a hierarchy between elements according to their relevance both temporally in a sequence and harmonically in a chord.

First, we have assessed our proposal by performing an ablation study whose results has highlighted the merits of each of the modules included in the model, in particular those of our HAM. Moreover, the whole system has outperformed the previous state-of-the-art prediction results demonstrating again the benefit of the embedding framework.

By displaying PCA projections of our embedding, we have further demonstrated that some musical features are reflected in the overall geometrical structure of the spaces. However, some examples do not respect these underlying patterns pointing to the fact that a musical chord can potentially carry very diverse "meanings" depending on the context in which it occurs. In broader terms, we can argue that there is a lack of meaning carried by a single musical object. Accordingly, trying to learn an embedding of musical events alone without any context (similarly to the approach taken in NLP) seems to be unrealistic.

Another important point of discussion is related to the relevance of the prediction task in learning musical embeddings. Indeed, in textual data, the order of appearance of different types of words is quite similar across the vast majority of sentences. In this context, learning to accurately predict future events turns out to be very beneficial for understanding the overall meaning of the language. On the other hand, in music, the ordering of the elements is less important and varies a lot depending on the different bars, which questions the overall validity of the prediction task for capturing meaningful features of music.

Furthermore, we faced the huge problem of orthogonalization, a recurrent phenomenon when training AE which consists in encoding the musical units in different orthogonal sub-planes in order to easily discriminate between them. Because of the contradictory nature of this process with the idea of learning a semantic

space, this issue has to be alleviate and the most direct way to do it seems to be through imposing constraints over the latent space during the training. Armed with these observations, we have developed in the following chapter another approach designed to improve the learning of symbolic embedding spaces.

4

VAE-BASED FRAMEWORK

4.1 INTRODUCTION

In this chapter, we propose another method for learning musical embedding spaces in the symbolic domain, which takes its roots from various observations collected during the evaluation of our first approach. To do so, we rely on the class of probabilistic AE called the VAE (see Section 2.3.2.5).

First, we detail the different choices that shaped our proposal and explain the underlying motivations behind our design choices. We present the architecture that we have developed and underline the critical problem of polyphonic music representation. Therefore, Section 4.3 is dedicated to this particular issue. We propose a novel representation called *signal-like* and conduct a benchmark against the three most successful representation of the literature. Then, we expose the results showing the efficiency of our proposal.

In order to assess the relevance of the resulting embeddings, we introduce in Section 4.4 a musical evaluation based on a synthetic dataset that follows strict music theory rules. We begin by describing the method by which we designed this dataset and, then, we outline its use in further analyses with their expected outcomes. Finally, we present the results achieved by 4 embeddings with these analyses, where each space is learned through a different input representation. This allows to ensure the consistency with the previous results and to measure the ability of the spaces to encode musical features of higher levels of abstraction.

4.2 MOTIVATION

Based on the mitigate success of the word embedding framework, we argue that there are some assumptions underlying its definition that are inappropriate for music processing. First, there are fixed ordering rules in textual data, which do not exist in music. Hence, defining learning through a prediction task appears less efficient in the music domain because of these less rigid ordering rules in music. Second, whereas we can define the semantic meaning of a single written word outside any context, there are no semantics in music without the corresponding context (time and rhythm). Therefore, it seems not possible to embed single

notes and chords, but we should rather focus on longer sequences instead. Finally, whereas there are very low amounts of repetition in text, repetition is a fundamental aspect in music.

Therefore, we have made several adaptations for further developing our symbolic embedding spaces. First of all, we do not consider single musical events as our analysis unit anymore but rather aim for small slices of music instead. Here, we face the somewhat controversial question of the choice of a "core unit" used to describe a musical score. If the granularity is too thin, the embedded elements will be too dependent on their contexts and, oppositely, if it is too large, the system will not be able to capture the subtleties contained in music. As discussed previously, the two most common alternatives found in the literature are either a fixed number of beats (usually 4) or several consecutive bars (between 2 and 16). However, the first case does not allow to handle pieces with different time signatures and the second case is more oriented towards the study of long term structures. Hence, in our context, by taking into account one bar of music (regardless of the number of beats), we hypothesize that the embedded objects can carry enough relevant semantic information by themselves, while keeping enough expressiveness.

Moreover, in order to have a finer control on the space properties, we now rely on probabilistic models, through the VAE, which allows to constrain the probability distribution over the latent space (see Section 2.3.2.5) and avoid the orthogonalization problem. Based on the success of the *MusicVAE* (Roberts et al., 2018) model, we rely on this architecture for our experiments. One of its strengths stems from its hierarchical decoder which divides the latent code into several sub-sequences allowing to manage much longer sequence. In our context, the temporal structure of our input data is shorter but the harmonic structure is much more complex since we are dealing with polyphony. Hence, we use a recurrent encoder with a two-layers bidirectional LSTM of 1024 units and a hierarchical recurrent decoder with a two-layer unidirectional LSTM with a hidden size of 1024 for both the conductor and decoder (Roberts et al., 2018).

Furthermore, we can see that the success of the wide variety of machine learning approaches rely on a crucial point: the input data representation. However, representing a polyphonic music piece efficiently in the symbolic domain appears to be a daunting task. The most well-known representation is the piano-roll, which accounts for the polyphonic nature of music but leads to highly sparse matrices that are notoriously inefficient for learning processes. Therefore, we focus on this particular issue of input representation in the next section.

4.3 POLYPHONIC MUSIC REPRESENTATIONS

The most commonly used format in music computing for polyphonic data is the *piano-roll* representation. This very sparse non-negative integer matrix describes note activations across time. However, the specificities of this piano-roll stirs up some issues for learning. First, the resulting matrices are high dimensional and the velocity is usually encoded with a categorical distribution. Moreover, because of the typically small amount of notes played simultaneously, these vectors are highly sparse. Hence, different symbolic music representations have been proposed to alleviate these limitations for machine learning approaches.

Motivated by this challenge, we introduce here a novel type of representation, which allows to encode any symbolic score as a minimal audio signal. Different MIDI pitches are mapped to prime frequencies and summed across time resulting in a simplified waveform. The process being perfectly invertible, we can retrieve the original score without losing any information. We show that this novel representation, named *signal-like*, is able to outperform previous propositions on polyphonic data. Hence, we compare our proposal to previous symbolic music representations, namely, the *piano-roll*, the *MIDI-like* and the *NoteTuple* representation by evaluating their results for learning musical embedding spaces.

4.3.1 *The signal-like representation*

An audio signal can be described as a sum of periodical functions, oscillating at different frequencies and complex amplitudes. Hence, we can represent any audio signal in a time-frequency domain such as the well-known *Short-Time Fourier Transform* (STFT), which produces a two-dimensional complex *spectrogram*. In this matrix, rows correspond to frequency bins while the columns describe each time step t (frames index). The advantage of the STFT is that it is an exactly invertible process, allowing to retrieve the original waveform from a spectrogram. In machine learning applied to music, this signal information offers several desirable properties. Indeed, it naturally contains polyphonic information as a decomposable sum. Moreover, there has been some large successes in using raw signal for learning. Nevertheless, the major flaws of this representation are its large dimensionality, and the existence of phase effects in harmonic signals.

In this section, we show that relying on a compact signal-like representation for polyphonic symbolic music can lead to large enhancements in tasks related to learning embedding. Hence, we aim to transform any given piano-roll as a spectrogram generating the most compact signal representation possible. To that

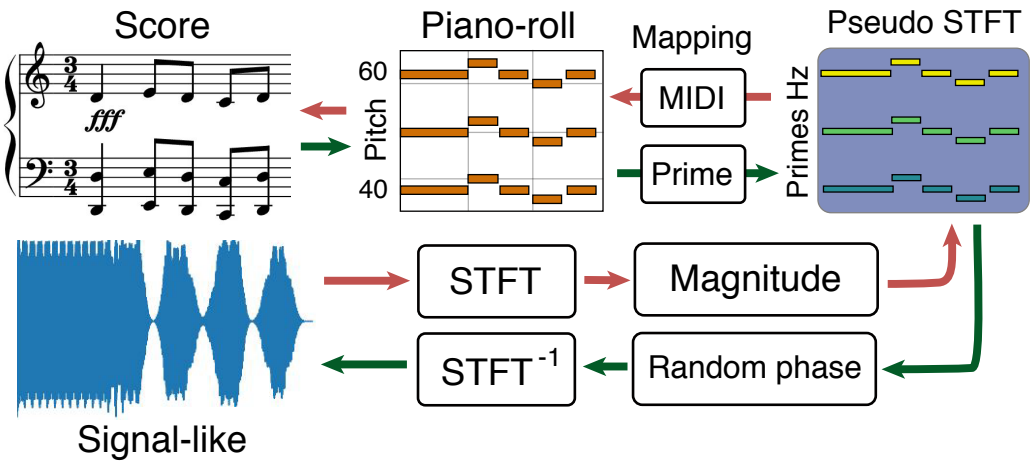


Figure 4.1: The score is represented as a piano-roll matrix where each MIDI pitch are mapped to prime frequencies. Then, taking the role of a fake audio signal phase, an imaginary part equal to 1j is added to each values of the matrix. Finally, the ISTFT is computed on the resulting spectrogram to obtain the final Signal-like representation

aim, we first map each MIDI pitch to prime numbers, starting from 43 (for MIDI pitch 0) to 2063 (for MIDI pitch 127) and removing consecutive numbers with a gap smaller than 4 (in order to not obtain pseudo frequencies that are too close together). In this representation, harmonic relationships may create phase effects detrimental to the inversion. Indeed, as the frequencies are arbitrary chosen and each represent a single MIDI pitch, the emergence of new harmonics during the inversion process would lead to the addition of notes which are not present in the original score. The use of prime numbers allow to mitigate this undesirable effect. Note that this also restrains the maximum frequency to a rather small value, allowing for a compact resulting signal. Then, we add an artificial phase, by setting imaginary parts to 0. Doing so, we can apply the inverse STFT to the resulting matrix and obtain a compact signal-like representation of the score. We use a window size of 2048 with an hop size equal to 1. This overall process is depicted in Figure 4.1.

4.3.2 *Benchmark*

In order to compare the efficiency of the *piano-roll*, the *MIDI-like*, the *NoteTuple* and the *Signal-like* representations, we evaluate them with the same common learning approach. Therefore, we train 4 identical models (with slight changes on the first

and last layer according to the input shapes) to provide latent spaces logically organized along music theory principles and compare the respective achievements.

4.3.2.1 *Experiments*

As discussed previously, we choose the hierarchical architecture of *MusicVAE* for our experiments. We evaluate different latent sizes in order to choose the most adequate value for largely reducing the input dimensionality while keeping enough information for the reconstruction.

We train the models on the Johann Sebastian Bach chorales dataset. We use data augmentation based on transposition as proposed in Hadjeres, Pachet, and Nielsen, 2017, ensuring that transposed chorales are still correct from a music theory standpoint. This leads to an extended set of 2418 training and 549 testing chorales. As the model takes musical bars as input, we split the MIDI files into separate bars and compute different representations on each bar. In order to obtain constant-sized matrices for both, the piano-roll and the signal-like, we slightly change the sampling frequency of the piano-roll according to the tempo.

We filter bars to have a maximum of 64 events, retaining more than 95% of all the data. After these operations, we obtain 36801 training and 8850 testing bars. Regarding the MIDI-like representation, we introduce a dummy event as padding to obtain a constant-size representation. We use a similar approach for the NoteTuple representation by taking 16 tuples by bar and padding with empty tuples.

All the models are trained with the ADAM optimizer (Chen et al., 2016) with an initial learning rate of 10^{-4} and a batch size of 16. In order to train the *piano-roll* and the *signal-like* representation, we rely on a MSE loss. Since the MIDI-like and NoteTuple representation are categorical, we rather rely on a cross-entropy loss.

4.3.2.2 *Results*

In this section, we compare the capacity of each model to reconstruct the original data, by relying on a frame-level accuracy measure. We divide musical sequences into elements of 16 frames and compute the accuracy as the ratio of correct (true positives) and wrong (false negatives and positives) active notes in each one of them. We also display separately the best reconstruction accuracy and KL divergence results on the test set in Table 4–3.

As we can see, the *MIDI-like* already provides strong performances in the reconstruction of monophonic musical bars but are unable to achieve the reconstruction

Table 4–3: Reconstruction accuracy and KL divergence on the test set for different input representations.

	Input	Reconstruction accuracy (%)	KL div
Monophonic	Piano-roll	95.8	$2 * 10^3$
	MIDI-like-mono	97.5	$1 * 10^3$
Polyphonic	Piano-roll	94.1	$2 * 10^3$
	MIDI-like	< 1	-
	NoteTuple	17.3	$9 * 10^4$
	Signal-like	**96.5**	$\mathbf{1 * 10^3}$

with polyphonic ones. Indeed, due to the nature of the *MIDI-like* representation, even a unique error on a `NOTE_ON` or `NOTE_OFF` event leads to an ill-defined musical sequences with notes that never end or never start. Regarding *NoteTuple*, the models are able to encode information about the number of notes, duration and time offsets, which are almost perfectly reconstructed. However, a large number of mistakes in the pitches of individual notes cause the frame-level accuracy score to be low. Hence, despite the use of regularization techniques (dropout, data augmentation) these two representations seem to largely suffer from over-fitting.

On the other hand, excellent results are achieved with the *signal-like* representation. In addition to a better reconstruction accuracy, our representation improves learning stability by avoiding the exploding gradient problem (Pascanu, Mikolov, and Bengio, 2012), which occurs with the piano-roll representation. Indeed, by training the model repeatedly using the piano-roll representation with various hyper-parameters, we have observed that below a certain KL divergence value, the gradient of the error becomes to low and cannot be back-propagated across the network anymore. As this behavior does not arise with the signal-like representation, our proposal decreases the reconstruction loss, while minimizing the KL divergence and, thus, leads to a better trade-off in scores.

Another interesting property lies in the robustness of the signal-like which is illustrated in Figure 4.2. In our particular setting, the inverse Fourier transform is not overly sensitive to minor variations in the waveform. In broader terms, we retrieve the perfect original score by applying our inverse process even on slightly noisy waveforms. Thus, small reconstruction errors inherent to any model are immplicitly erased and do not alter the output data, whereas for piano-rolls this automatically leads to added or deleted notes in the resulting score.

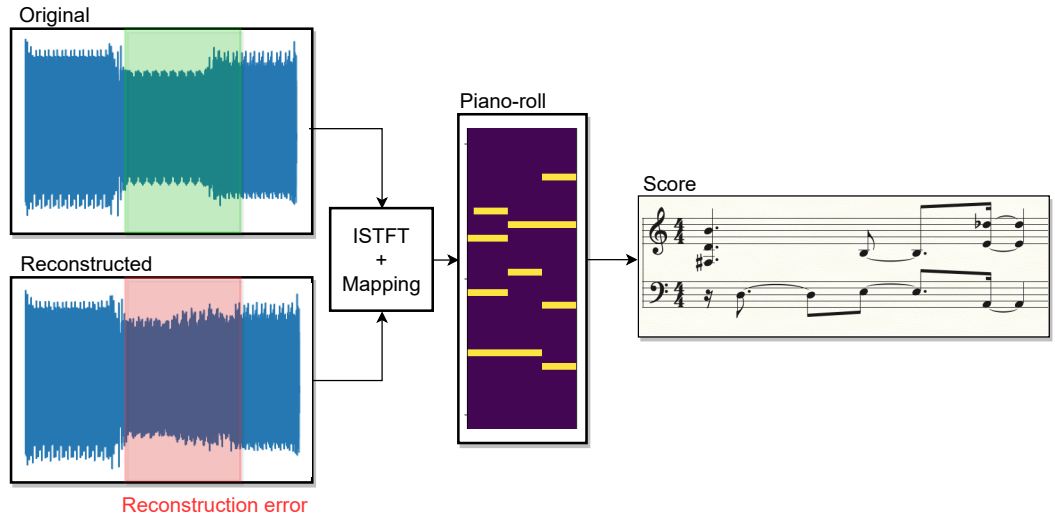


Figure 4.2: Example of a reconstructed signal-like representation. Despite the fact that the reconstruction error slightly blur the waveform, the corresponding piano-roll and final score are perfectly retrieved.

EMBEDDING DIMENSIONALITY Here, we assess the number of dimensions needed for the spaces to achieve good reconstruction results, while maximizing the compression and minimizing the KL divergence. The results showing the impact of the dimensionality on the reconstruction accuracy and KL score are shown in Figure 4.3. As we can see, 256 seems to be the best value in the signal-like context. Note that, for the Piano-roll, the system allows a better compression because the dimensionality can be decreased to 128 without significant loss of information. This seems logical given the fact that the space retains globally less information in this case, as shown by the reconstruction and KL divergence scores.

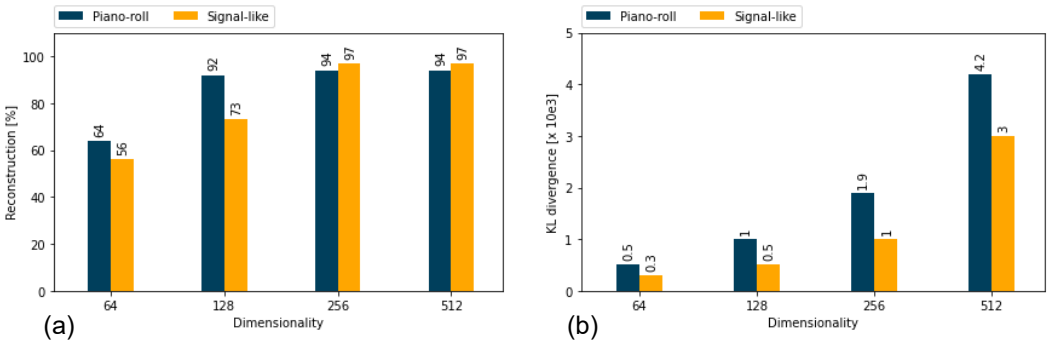


Figure 4.3: Impact of the latent space dimensionality for the piano-roll and signal-like representation. (a) Results for the frame-level reconstruction accuracy. (b) Results for the KL divergence with a factor of 10^3.

4.4 SPACES EVALUATION

In this section, we address another key issue in learning symbolic music embedding, which lies in their evaluation. Indeed, although all embedding spaces inherently compress the input information into a small set of features, the relevance and organization of these different sets of embedded features are non trivial to compare. These features can represent different levels of abstraction, and might also be highly context-dependent, varying between different genres or even within a single musical piece.

To address this issue, we introduce an evaluation method based on principled music generation, following known rules from music theory. Hence, to allow for precisely controlling the properties of evaluation data, we focus in this chapter on the four part harmonized chorales by Johann Sebastian Bach, which follow clear and well-known music theory rules. Our method allows to create synthetic musical bars (termed *skeletons*), following the rules of Bach chorales. From each skeleton, we create a set of sophisticated pieces by adding passing tones, neighboring tones, suspensions and retardations while keeping the previous rules respected. Finally, we compute the correlation between the learned embedding and our pre-defined music theory properties, allowing to quantify how different spaces are organized along these musical concepts. In the next section, we show the superiority of our *signal-like* representation to learn well-organized music embeddings.

4.4.1 *Musical analysis*

In order to precisely evaluate how embedding spaces are able to capture music theory principles, we introduce a principled generative approach. This allows to obtain large sets of evaluation data with controlled properties, while following known music theory rules similar to the ones applied in the four-part harmonized chorales written by J-S Bach.

4.4.1.1 *Principled music theory dataset*

Hence, we generate a dataset composed by bars of synthetic chorales in *C major* or *a minor* tonality. We follow the strict modulation rules implemented by Bach in his chorales and allow only the use of neighbouring tonalities (G, F, e, d in our case). Bars are either in one of these six tonalities or modulate between them. Finally, we generate the data by random sampling with the following procedure

1. We generate sequences of tonal functions with first-order Markov chains, with transition probabilities defined by expert composers.

2. We expand this sequence as a four-voices realisation, by randomly picking chords in major triads, minor triads, diminished triads and dominant sevenths. This defines a *skeleton* of only quarter notes.

3. Based on the *skeleton*, we generate more sophisticated realisations by using passing tones, neighboring tones, suspensions and retardations without fundamental changes in the chord progression.

In order to further control the generation properties, we define two sets of rules related to the harmony and voice progressions. These rules allow to

1. define prior constraints on sampling through
    - voice pitch ranges
    - rules on double and missing notes in chords.
2. exclude erroneous samples containing
    - bad transitions of the seventh and leading-tone.
    - parallel octaves, fifths and unison
    - direct octaves, fifths and unison
    - unisons left by direct motion
    - unisons approached by oblique motion

Finally, we obtain a set of 21966 realisations from 370 different skeletons, where the links between each realisation and its corresponding skeleton have been kept. In our experiments, this corpus is used only for evaluation purpose. Hence, none of these data are used during the training.

Based on this corpus, we can analyze different aspects of the organization of learned embedding spaces. First, we can analyze the behavior of different spaces based on tonality. An adequate clustering of the different tonalities across the space would prove its capacity to encode this high-level musical information. Second, we can compute the distance between a realisation and its corresponding skeleton based on the *number* of non-harmonic tones. In this case, if the space is well-organized, we expect these distances to evolve linearly with the number of additional tones. Third, we can compute the distance between a realisation and its corresponding skeleton based on the *type* of non-harmonic tones. A link between these distances and the note type would show that the embedding space can handle advanced musical concepts. Finally, we can compute the distance between consecutive skeletons. Since a realisation is always musically closer to its corresponding skeleton than to a different one, this condition should be reflected in the metrics of a well-structured space.

4.4.2 *Results*

First, we embed unseen synthetic bars in order to compare their respective projections in the embedding. As the latent spaces have 256 dimensions, precluding direct visualization, we rely on a t-SNE dimensionality reduction approach (Maaten and Hinton, 2008a) with 1000 iterations and a perplexity of 30. We display in Figure 4.4 the results of this analysis according to either 3 different tonalities (a) or to 6 different tonalities (b).

It clearly appears that the use of our proposed *signal-like* representation significantly improves the efficiency of the latent space in encoding an unseen high-level musical feature such as the tonality. Indeed, in the case with 3 tonalities, the t-SNE projection of the embedding shows that input sequences are organized along clean clusters with smooth transitions between them. For the piano-roll, the resulting space appears to suffer from a lack of smoothness despite the fact that the tonalities have been well separated. Finally, as expected from the reconstruction scores, there is no strongly logical organization of the spaces learned through the NoteTuple and the MIDI-like representations. These results are also reflected in the analysis with the 6 tonalities where the separation is not as clear – potentially arising from errors within the mapping of the t-SNE algorithm, which struggles to entirely express the complexity of the organization along the 256 dimensions – but still remaining very apparent in the signal-like case.

Then, we compute statistics over metadata of the toy dataset in order to evaluate how different spaces have organized the realizations with respect to the number of non-harmonic tones. As we can see in Figure 4.5, the distances between the original skeleton and its different realizations in the space learned through the signal-like representation provide an almost linear relationship with the number of non-harmonic tones, whereas for other representations, this function is even not monotonic. This highlights the fact that the *signal-like* space is better organized from a music theory point of view.

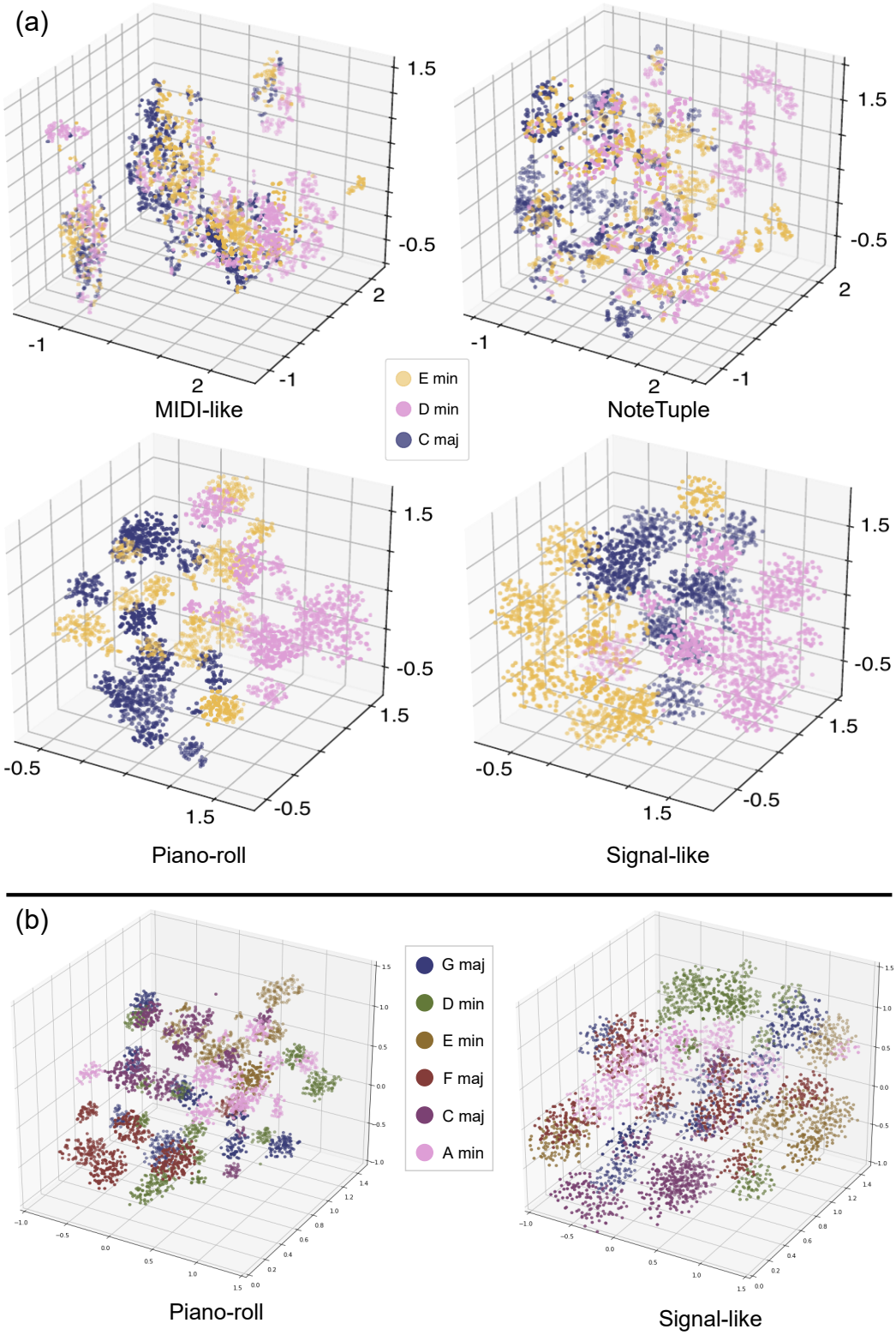


Figure 4.4: t-SNE plots of different latent spaces with our principled synthetic data, where these musical bars have not been seen during training. The colors represent the tonalities ((a) with 3 tonalities and (b) with 6 tonalities). We use a perplexity of 30 and 1000 iterations.

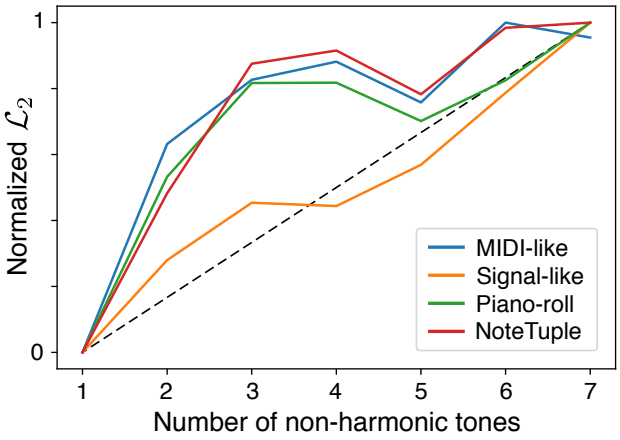


Figure 4.5: Normalized mean $\mathcal{L}_2$ distances between the original skeleton and its realizations depending on the number of non-harmonic tones.

In addition, we compute the $\mathcal{L}_2$ distances between each skeleton. This distance should be larger than the ones between a skeleton and all of its realizations in order to imply a coherent organization of the latent space. Indeed, regardless of how many notes have been added to a skeleton, realizations are always closer musically speaking to their skeleton than two totally different sequences. As we can see in Table 4–4, this condition is well respected in the case of the signal-like and the piano-roll representation.

Input	DBSR	DBCS
Piano-roll	229.0 ± 36.0	291.7 ± 20.4
MIDI-like	445.5 ± 169.1	312.8 ± 92.6
NoteTuple	572.5 ± 146.5	292.8 ± 89.4
Signal-like	242.2 ± 34.2	285.5 ± 13.5

Table 4–4: Distances between the original skeleton and its realizations (DBSR) and distances between consecutive skeletons (DBCS).

However, it seems that none of the spaces have been able to correctly encode information regarding the *type* of non-harmonic tones. Indeed, we observed the distance between a skeleton and its realizations with only one non-harmonic tone depending on its type. The corresponding behavior did not reflected any rules for the *MIDI-like* and the *NoteTuple*, while there were very similar for the two other representations (underlining that their metric properties are highly tied to the number of alterations).

4.5 CONCLUSION

In this chapter, we have explored another approach for learning musical embedding space targeting the main weaknesses of the first method presented in Chapter 3. First, in order to obtain more meaningful embedding vectors, we have considered the basic musical unit to be an entire bar rather than isolated musical events. Second, by relying on the latent space of the VAE, we have performed a control over the final embedding thus preventing the orthogonalization problem. And, finally, we have avoided the prediction task in the training procedure to rather focus on the reconstruction of the encoded data.

Given the success of the *MusicVAE* model to learn musical embedding for monophonic melodies, we have used a similar architecture under the assumption that its hierarchical decoder designed to manage long-term temporal structures could be also beneficial for decoding complex polyphonic structures. Moreover, we have seen that the achievement of this system is highly bound to the input representation. Since the piano-roll exhibits poor statistical properties, we have proposed a new representation for polyphonic symbolic music, called *signal-like*, which transformed a small excerpt of music in a pseudo-waveform naturally containing polyphonic information. By conducting a benchmark against the piano-roll, MIDI-like and NoteTuple representations, we have shown that our proposal have improved the quality of the learned spaces for both the reconstruction quality and the organization of the underlying spaces.

Finally, as the evaluation of such spaces from a musical point of view is a daunting task, we have proposed a new musical evaluation for latent spaces with a specifically tailored synthetic dataset. We provided a generation procedure, allowing to create synthetic bars with well-defined and controled music theory properties. In addition to having again evidenced the efficiency of our representation, these results have highlighted a musical coherence in the structure of our embeddings, providing a large step forward towards our objective.

5

APPLICATIONS

5.1 INTRODUCTION

In addition to providing an efficient input representation for complex models, our embeddings learned through VAE can be employed directly for creative or analytical purposes. In this chapter, we present some applications, which aim to either infer knowledge from musical data or enhance composers' creativity.

In Section 5.2, we present a composers classification tool relying on the meaningful structure of our embeddings. After presenting the experiment, we highlight the results of our model to classify small musical sequences between 5 notable composers of Western music. Besides, while discussing the success of this approach, we also provide the results of the binary classification task defined by the discrimination between sequences from one composer and all the other composers.

Afterwards, Section 5.3 is dedicated to creative applications. Hence, we introduce experiments in *attribute vector arithmetic*, which intends to modify a musical bar in a meaningful manner. We further present the results of interpolation and latent code averaging between two distinct sequences. Lastly, we display several plots that illustrate most of the proposed methods and discuss the results.

Finally, we conclude this chapter by providing a summary and an overall discussion of our various proposals.

5.2 COMPOSERS CLASSIFICATION

In this section, we propose a direct application of our embedding spaces, which aims to classify small musical score excerpts according to their composer. As writing music is a very intricate process, which reflects the personal vision of the composer, we expect the meaningful structures of our embeddings to aid in distinguishing between their different styles. Moreover, such a tool can be of particular benefit for inferring musical knowledge about a composer or musical trend, as well as for performing tasks such as automatic labeling of datasets.

Composers	Train	Test	Valid
Scriabin	19	4	5
Debussy	31	6	4
Scarlatti	18	5	4
Mendelssohn	27	4	0
Liszt	99	17	13
Schubert	150	24	6
Chopin	159	21	18
Bach	129	10	10
Brahms	20	2	2
Haydn	25	3	10
Beethoven	104	9	21
Schumann	28	7	8
Rachmaninoff	46	10	0
Mozart	30	3	2
Others	53	0	0

Table 5–5: Musical pieces distribution over the composers of the MAESTRO dataset (Hawthorne et al., 2019).

5.2.1 *Settings*

First, we train our previously introduced system (relying on a VAE architecture and our signal-like representation) on the MAESTRO dataset proposed in Hawthorne et al., 2019. This dataset contains over 200 hours of paired audio and MIDI recordings from piano performances of classical pieces written by composers from the 17^{th} to the early 20^{th} century. In our context, we used only the MIDI files, split as recommended by the authors into train, test, and validation sets – described in Table 5–5 – leading to a total of around 117K training bars and 23K for testing and validation purposes, each represented through the signal-like representation.

Once our model has reached the lowest possible KL divergence value while keeping a reconstruction accuracy score above 96% on unseen datasets, we stop the training and keep the encoder with frozen parameters to project data in the resulting embedding space. Then, we use the embedded samples as input representation for training a 2-layers RNN classifier with 256 units per layer, which intends to segregate small musical sequences based on their composers. As the MAESTRO dataset is highly unbalanced and contains only few examples for a large

Composers	Train	Test	Accuracy
Bach	3088	553	84%
Beethoven	6055	797	54%
Schubert	7428	1017	46%
Chopin	6027	1367	46%
Liszt	5082	485	77%
Total	27680	4219	58%

Table 5–6: 6-consecutive bars sequences distribution over the 5 most represented composers of the MAESTRO dataset. In addition, we display the results of the classification task on the test set.

variety of composers, we extract a subset comprising pieces written by the 5 most represented composers and split them in sequences of 6 consecutive bars *without any overlap* (otherwise, the network simply tries to detect common bars among the sequences and thus heavily suffers from overfitting) that have been first encoded in the embedding space. Finally, we obtain a set of 27680 sequences for training and 4219 for testing. We display the categorical distribution of the data, as well as the classification results in Table 5–6.

5.2.2 *Discussion*

As we can see, the overall classification results are fairly good in this context. As a reference, a similar model trained without going through an intermediate embedding representation or through an ill-defined embedding does not exceeds 30% accuracy for all the composers which is hardly better than the random function. Now, the most interesting part of these results is the variation in the model accuracy among composers. Indeed, although the model has trouble classifying almost half of the sequences composed by Beethoven, Chopin, and Schubert, it performs greatly in identifying the work of Bach and Liszt. Thus, by precisely analyzing the musical relationships between correctly classified sequences in the embedding, such a system can be an interesting tool for musicologists to refine our knowledge on the compositional practices of different composers.

On the other hand, the classifier is not sufficiently efficient by itself to do automatic labelling on these 5 composers simultaneously. Nevertheless, as it seems able to distinguish the work of certain composers, it can potentially be used to dif-

Composers	Accuracy
Bach	91%
Beethoven	86%
Schubert	69%
Chopin	61%
Liszt	89%

Table 5–7: Results on the binary classifications aiming to differentiate a work of a composer among the others.

ferentiate a work of a given composer amongst all others and thus operate a class by class labeling. In order to show this, we have trained the same architecture with the same data to perform this binary classification task for each composer. The accuracy is expressed as a ratio of good predictions defined by

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{FN} + \text{FP} + \text{TP} + \text{TN}} \tag{5–56}$$

with TP, TN, FP, FN the true positives, true negatives, false positives and false negatives predictions respectively. As we can see in Table 5–7, the resulting scores are rather high despite the fact that, as expected, the task is more difficult for certain composers than others. However, in the high accuracy cases, the results are strong enough (notwithstanding a remaining amount of errors warranting a manual check for the less reliable predictions), such a system can be an adequate basis for automatic labeling.

5.3 CREATIVITY SUPPORT TOOL

Thanks to the probabilistic nature of the VAE, our model can generate new realistic data by performing sampling in the latent space and decoding the resulting latent codes. In this section, we use this property to propose several creative tools, which aim to enhance the composition process or to serve as a basis for more complex systems. All of the following experiments have been done using our model trained on the MAESTRO dataset (previously introduced in Section 5.2).

5.3.1 *Attribute vector arithmetic*

First, we propose an application that aims to alter major musical features (or *attributes*) of a given bar in a controlled manner. Indeed, by averaging the latent code of a given class (i. e.with a common attribute) of musical data, we can obtain

a so-called *attribute vector*. We can see these as the unit vectors of the embedding space along which the value of the corresponding attribute will vary. Hence, by performing simple arithmetic operations, the attribute vectors can be used to remove or add the corresponding attribute to a musical bar.

As shown in Raffel and Ellis, 2016, MIDI files are sorely lacking in robust annotations, prompting us to resort to attributes that can be trivially computed from the score itself. Hence, we rely on the 6 following attributes.

- **C diatonic membership** : The amount of notes in a bar which belong to the C diatonic scale (i. e.the white keys of a piano C, D, E, F, G, A, and B). This attribute is expressed as the ratio of these notes to the number total of notes, thus taking values in the range $[0, 1]$.
- **Note density** : The number of notes played in a bar.
- **Average polyphony** : The mean number of notes played simultaneously across a bar (i. e.the average length of the chords)
- **Average note duration** : The mean of the duration of the notes in the bar.
- **8^{th} note syncopation** : The ratio of quantized note onsets which are not landing on an expected 8^{th} note position without any note played in the previous 8^{th} note position.
- **16^{th} note syncopation** : The ratio of quantized note onsets which are not landing on an expected 16^{th} note position without any note played in the previous 16^{th} or 8^{th} note position.

To extract the attribute vectors, we apply a specific procedure described in the following. First, we compute the attributes for each training sample and partition them into quartiles based on their corresponding value in terms of attribute. Then, we compute the attribute vectors by subtracting the mean latent vectors of the top quartiles to the mean latent vectors of the bottom ones.

Once equipped with these vectors, we first measure the adequacy of the semantic modifications that can be drawn from them. To do so, we start by measuring the amount of attributes expressed by a total of 256 bars, which have been formerly generated by random sampling from the prior. Next, we replicate this procedure on the bars resulting from the decoding of the latent codes obtained through the additions and subtractions of the attribute vectors to the 256 original codes. Finally, we assess the average percentage of change that are obtained after applying these different operations.

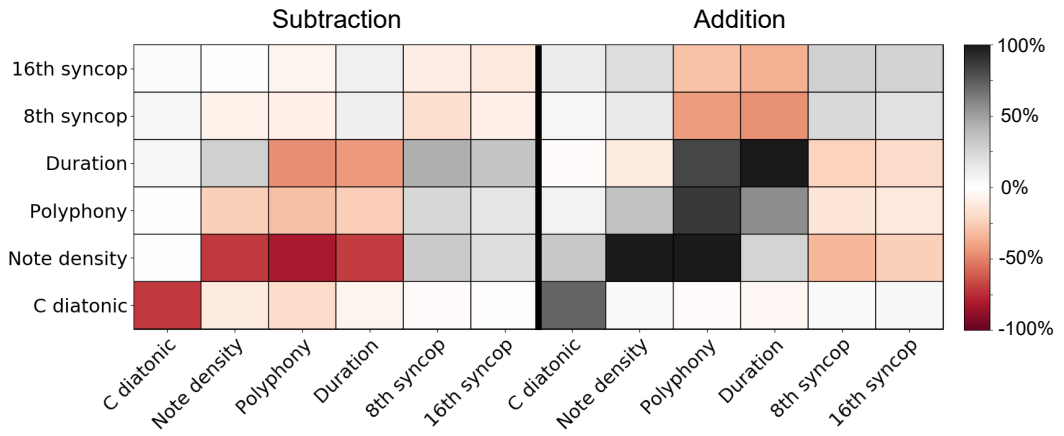


Figure 5.1: Percentage of change in the attributes induced by performing attribute vector arithmetic. The diagram on the left illustrates the subtraction, while the right depicts the addition.

As shown by the results displayed in Figure 5.1, the attribute vectors allow significant changes in the expression of the corresponding attributes and, thus, produce the desired semantic shifts. However, these transformations often go hand in hand with others that were not initially intended. In some cases, these side effects are mostly logical as they involve attributes that are highly semantically related such as the average polyphony and the note density. Conversely, some of these relationships are somewhat not so easily established, but given the broad nature of our attributes, it seems natural that some of their characteristics are overlapping. Thus, we assume that this method could be greatly improved by specifying finer attributes based on a more thorough musicological analysis of the data. That being said, with regards to its performance and the control it allows, the tool introduced here still holds a great deal of creative potential.

For a more concrete understanding of the impact of the attribute vectors arithmetic, we have provided several plots illustrating the contrast between original and altered bars in Figure 5.2, Figure 5.3, and Figure 5.4. Since we also sought to evaluate whether we could finely control the amount of the targeted attribute, we have multiplied the attribute vectors by a factor whose values vary between -1.5 to 1.5, in steps of 0.5.

As we can see in these various examples, the results observed in Figure 5.1 are clearly reflected in the scores. Indeed, it appears that the C diatonic is the most successful case with a clean and smooth modification of its attribute without significant side effects even for the most extreme values. The note density attribute also shows great performances, as evidenced by this particular example, where

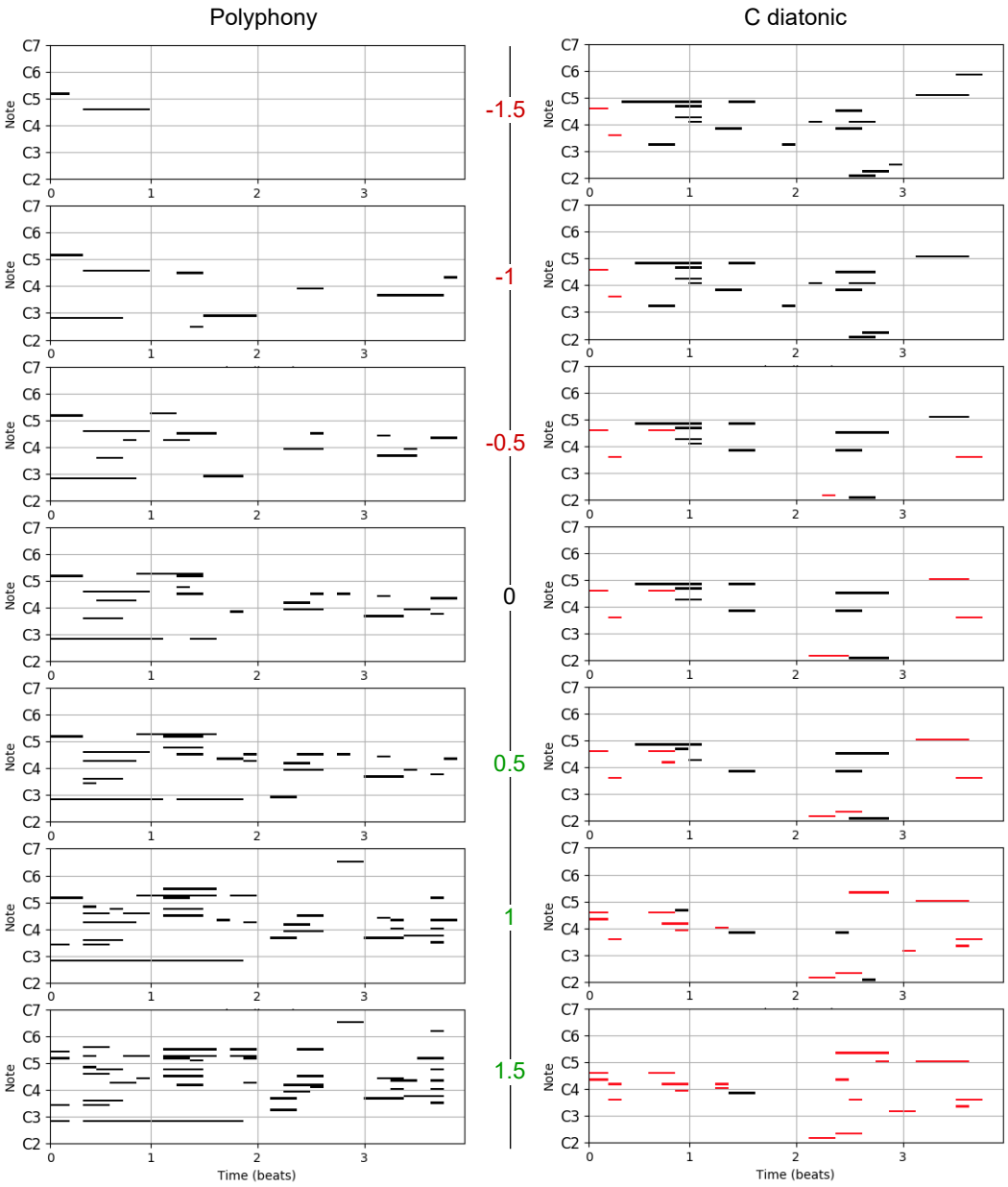


Figure 5.2: Examples of attribute vector arithmetic performed with the C diatonic membership and the average polyphony attributes. The unaltered bars have been generated through random sampling from the prior distribution and the attribute vectors have been multiplied by the factor depicted on the scale in the middle before being added to the original latent codes. For the C diatonic attribute, the notes which belong to this scale have been colored in red to ease the visualization.

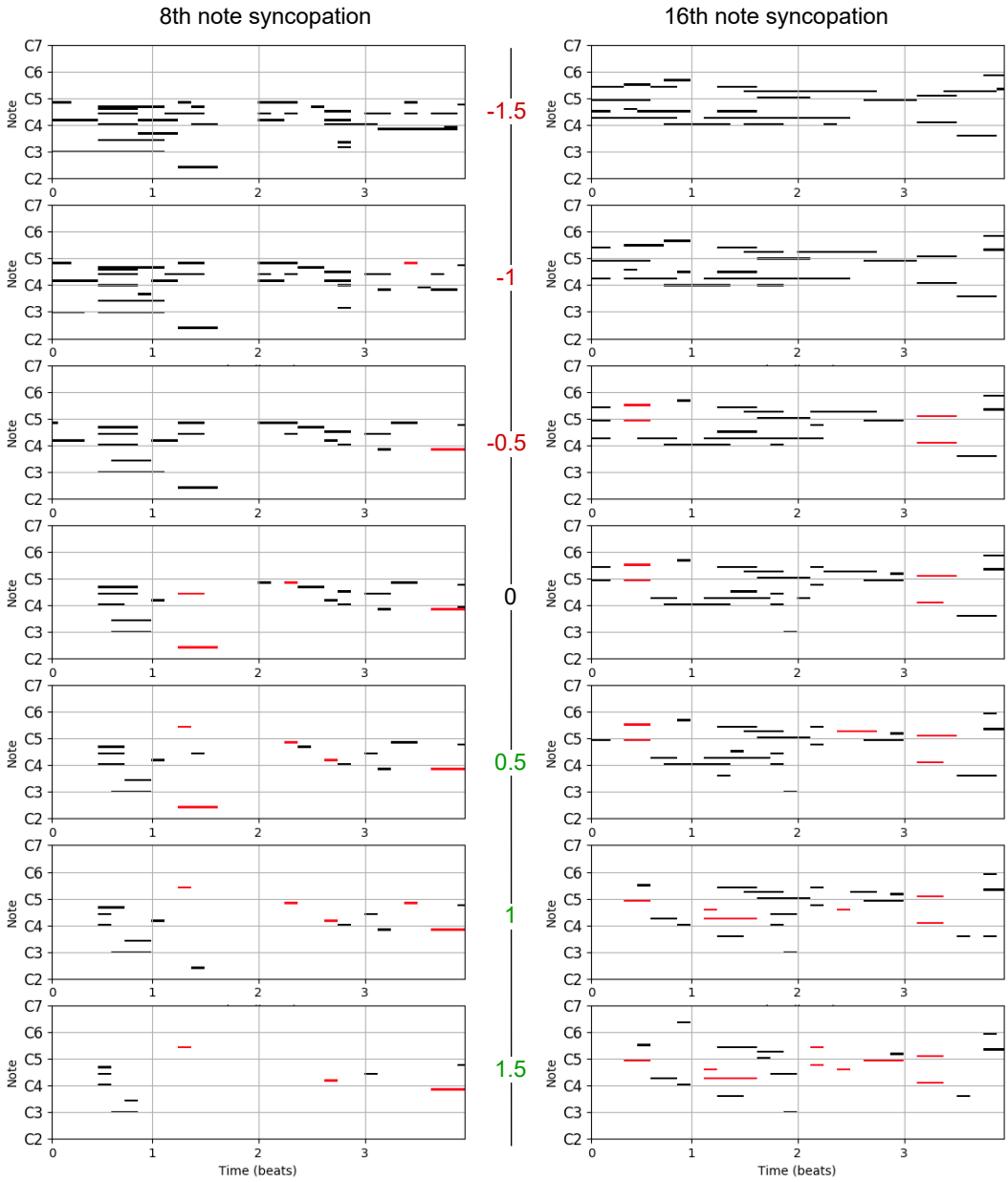


Figure 5.3: Examples of attribute vector arithmetic performed with the 8th note and 16th note syncopation attributes. The unaltered bars have been generated through random sampling from the prior distribution and the attribute vectors have been multiplied by the factor depicted on the scale in the middle before being added to the original latent codes. The note with syncopated onsets have been colored in red to ease the visualization.

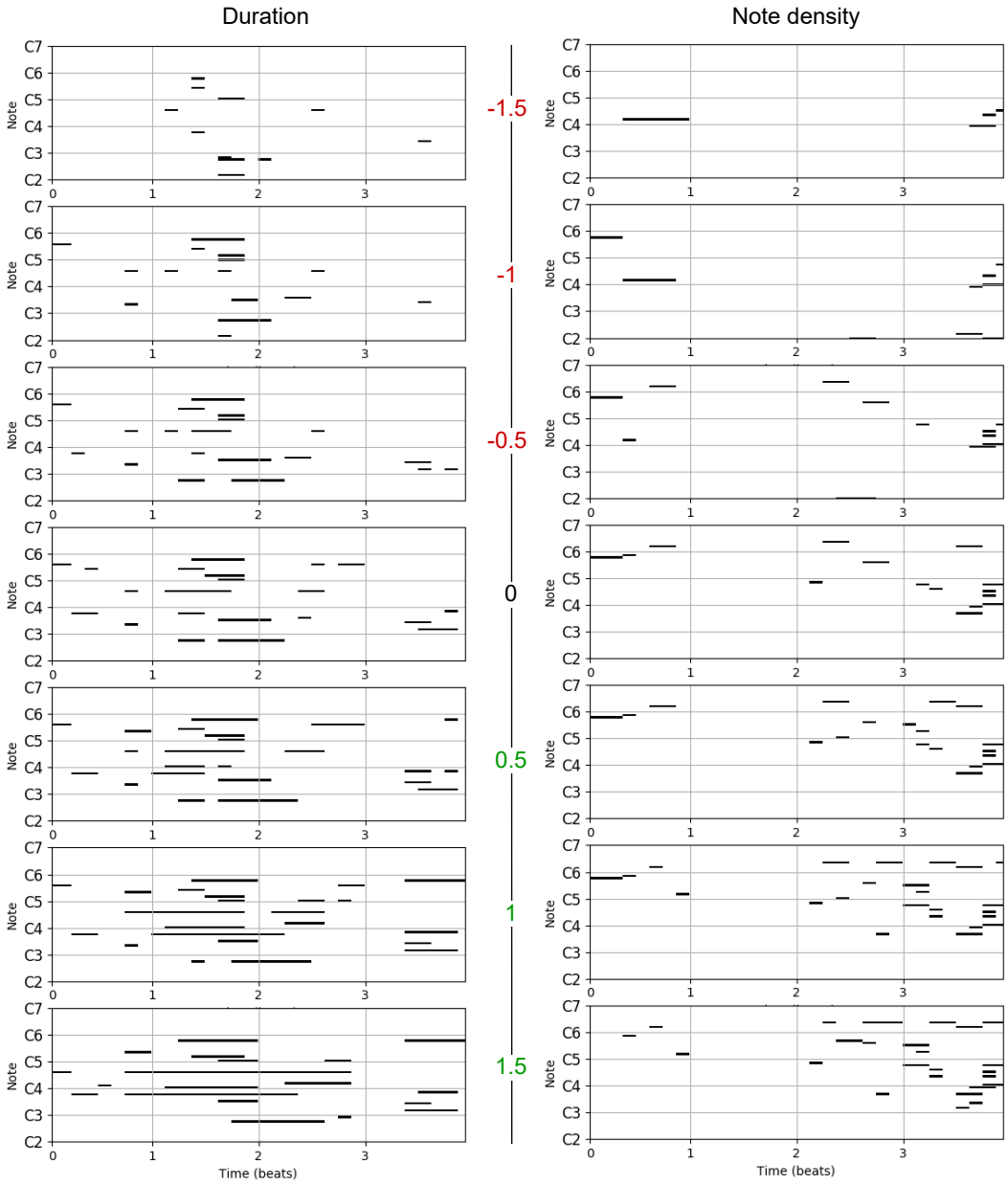


Figure 5.4: Examples of attribute vector arithmetic performed with the average note duration and density attributes. The unaltered bars have been generated through random sampling from the prior distribution and the attribute vectors have been multiplied by the factor depicted on the scale in the middle before being added to the original latent codes.

the attribute has been correctly modified without modifying the structure of the original bar characterized by a rest on the 2^{nd} beat. Regarding the polyphony attribute, even though it is closely related to the density as expected, the desired variations are also very significant. Finally, although still noticeable, the effects of the syncopation attributes are the most mitigated since the expected modulation remains quite low, but with a significant impact – especially for the 8^{th} note case – on the average duration and density.

5.3.2 *Interpolation*

To further highlight the generative capabilities of our system, we have made several experiments relying on interpolations. Indeed, the continuity and the organized structure of the latent space should allow to generate smoothly varying bars from the interpolated latent codes between two points. Thus, by computing

$$e_\alpha = \alpha z_1 + (1-\alpha) z_2 \tag{5–57}$$

with $\alpha \in [0,1]$, z_1 and z_2 the latent codes of two points x_1 and x_2 respectively. By decoding the M resulting e_α, we should obtain realistic musical bars x_{e_α} for each α, with x_{e_i} semantically closer to $x_{e_{(i+1)}}$ than $x_{e_{(i+2)}}$ for $i \in \mathbb{N}^M$. However, as pointed out in the literature, the underlying metric of the Gaussian prior of a VAE is spherical rather than linear, thereby making spherical interpolation – with a change on the radius – a generation process, which is more consistent with the inherent regularization of the VAE. Thus, in our experiments, we rely on the spherical interpolation formula, defined as

$$e_\alpha = \frac{\sin\big((1-\alpha)\Omega\big)}{\sin(\Omega)} z_1 + \frac{\sin(t\Omega)}{\sin(\Omega)} z_2 \tag{5–58}$$

with $\cos(\Omega) = z_0 \cdot z_1$.

We display in Figure 5.5 some interpolation examples, where we have selected random testing samples and generated the bars corresponding to the interpolated latent codes between them. In these figures, the first and last bars are the random elements of the testing set, while the others are generated by our system. As we can see, the generated sequences are gradually evolving from one sample to the next, even though the characteristics of the two original bars are very different. Note that the rate of change in the generated bar sequence can be controlled by varying the size of the interpolation step and, thus, generating more or fewer bars between the start and end points. However, if the step size becomes too large, the evolution might lose its smoothness and if it gets unnecessarily large, some of the consecutive generated bars will be roughly identical. Thus, if we want to

obtain steady results regardless of the original data, a potential solution consists in setting this step size according to the distance computed between the starting and the ending bars in the embedding space.

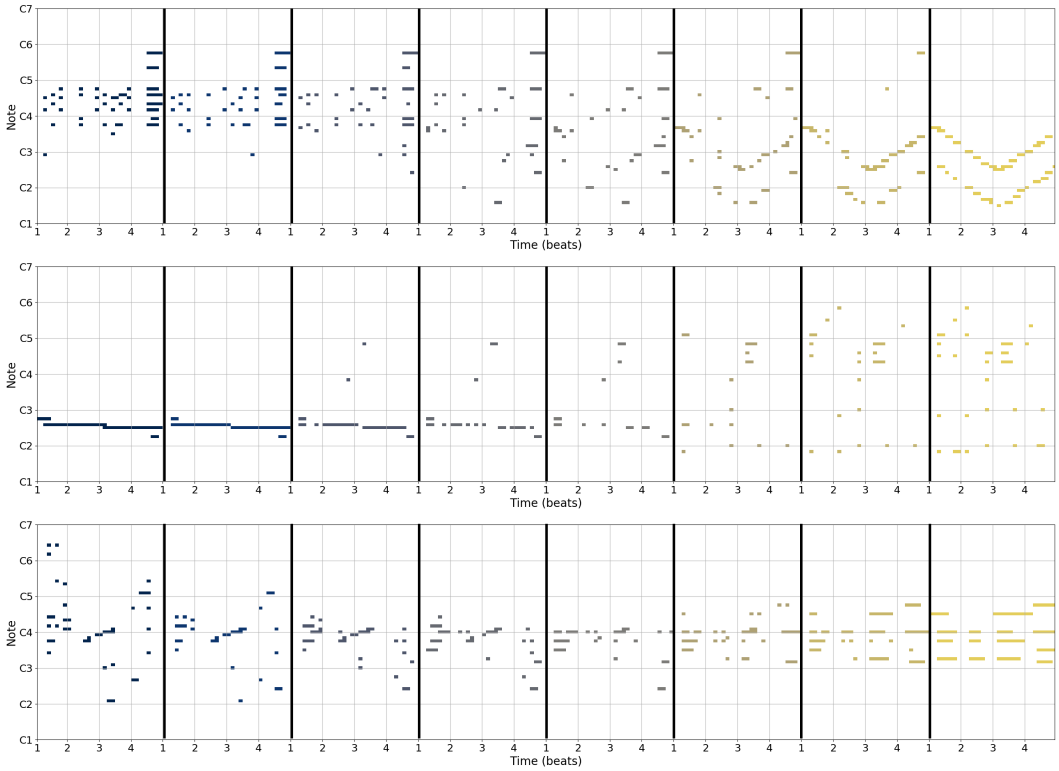


Figure 5.5: Interpolation generation between randomly selected training points. The first black and final yellow bars are the original training samples, while the others are generated by our system. The interpolation step was chosen to generate 6 bars between the starting and finishing points.

5.3.3 *Discussion*

The different results presented in this chapter underline the creative potential of our embedding spaces. Indeed, while providing realistic musical data, the manipulation of latent codes allows a degree of semantic control over the resulting bars, by intentionally modifying their core attributes. In this context, we believe that our embeddings can provide a powerful tool to enhance composers creativity by proposing meaningful alterations of a selected bar through attribute vector arithmetic or close neighborhood latent codes sampling. In addition, such a system could provide a "mix" between two distinct bars by averaging their respective latent codes. In that way, we assume that our tool will not tamper with the creative intent of the artist but rather improve the compositional experience by providing

rapid access to a palette of possible alternatives. However, we keep in mind that to fully exploit this creative power, it is necessary to refine the definition of the musical attributes by conducting a finer analysis of the training samples.

Regarding the interpolations, our system also achieves interesting results by generating bars that are smoothly varying from one sequence to another, although they are musically very far apart. While this nicely illustrates the structural properties of our embeddings, the creative potential of such a process seems rather limited when based on basic linear interpolations. However, the possibilities for various spatial progressions are endless – such as advanced interpolations, path distortions, shape designs, and so on – and we believe that exploring them is a creative process in itself that can lead to highly interesting results.

5.4 CONCLUSION

In this chapter, we have introduced several applications of our embeddings learned through the signal-like representation and the hierarchical VAE. We have trained the model on the MAESTRO dataset, which gathers the reductions for piano of a great number of orchestral pieces from Western classical music. In addition to containing proper MIDI files, this dataset has the benefit of being more musically diversified than those presented in the previous chapters, while remaining within the restricted scope of Western classical music.

First, we have presented two composers classification tools, which are performing fairly well in categorizing the work of 5 great composers of the Western classical music. Although these performances are rather unequal depending on the composer, they are, nevertheless, very promising given the simplicity of the system implemented. Indeed, as our goal was to demonstrate the benefits of our intermediate latent representation, we have not performed extensive optimisation of the classifier hyper-parameters, nor of the training procedure. Moreover, by means of a simpler task consisting in discriminating between the sequences from a given composer and the others, we have succeeded in producing a very accurate tool.

Then, we have proposed a more creative use of our embeddings through several examples of latent code manipulation that have yielded meaningful variations on the bars. Indeed, by categorizing the training samples according to a set of attributes, we were able to compute the corresponding attribute vectors that can be considered as the most relevant latent dimensions involved in encoding these features. Thus, by adding or subtracting these vectors to the latent code of a bar, we can purposefully alter the amount of attribute exhibited in it. While our results

have confirmed that this technique produces the expected semantic shifts, we have also observed that it has an impact on other attributes that were not initially involved. To avoid this side effect, we assume that the definition of the attributes has to be refined to not cause overlapping in our proposed heuristic.

Finally, we have confirmed the generative capabilities of our system by displaying various plots illustrating sequences generated from linear interpolations or average latent codes between two distinct embedded units. Since this process produces realistic musical sequences that are smoothly evolving, it demonstrates, once again, the consistency in the underlying music theory structure of our embeddings that empowers them with a genuine creative potential.

6

CONCLUSION

To conclude our work, we start in Section 6.1 by summarizing all of our contributions presented in each chapter of this manuscript. Then, we present in Section 6.2 the various axes for future work arising from our research. Finally, in Section 6.3 we provide an overall conclusion on the work conducted throughout this thesis.

6.1 SUMMARY AND DISCUSSION

First, in chapter Chapter 1, we introduced the manuscript by highlighting the challenges raised by the representation of symbolic music in the context of computer science and how *musical spaces* offer a powerful framework for addressing it. We set the background of this thesis and explained our motivations in exploring the use of machine learning algorithms for learning meaningful unsupervised low-dimensional spaces called *embeddings*. Therefore, obtaining these types of spaces for symbolic music data was the guideline of this thesis.

In Chapter 2, we made an extensive overview of the state of the art related to our research. We provided a quick history of musical notation, underlining its role in the development of music and presented the main proposals found in the literature to allow its use in computer-based processing. Along with the respective benefits and drawbacks of music representations, we have seen that these are far from entirely satisfactory. We also exhibited the development of musical spaces, starting from the first circular representation of pitches to the most recent versions of the *Tonnetz*, which have highly motivated our research. Next, we have provided an in-depth presentation of the concepts of machine learning from its basic definition towards the most advanced models we used in order to achieve our objectives. Then, we explained how this framework has led to the emergence of *embedding* spaces, as well as the tremendous breakthroughs that they triggered in the NLP field. Finally, we presented different approaches related to our work, which aims to adapt this paradigm to musical data. Similarly to our manuscript, we have organized these proposals into two categories, namely those relying on *event prediction* and those relying on the *latent space* of probabilistic models.

Chapter 3 was dedicated to our first method derived from the NLP field. Starting out from the observation that there are some critical features in musical symbols

that do not exist in text (such as the notion of pitch class and the important temporal dependencies defined by the rhythm), we proposed a new model designed to fill these gaps through a CNN and a LSTM module. In addition, we enhanced the overall performances of our system by introducing a novel attention mechanism capable of distinguishing the harmonic and temporal salience of elements in a sequence at all levels of abstraction (i. e.layers) of the network. Our proposal was able to overpower all previous state-of-art results on the prediction task. Moreover, by using the t-SNE algorithm to project our high-dimensional embedding spaces into 2-dimensional maps, we have revealed the existence of geometric relationships between objects that exhibit an understanding of musical semantic concepts. However, since there were no constraints set on these spaces' properties, the model naturally encoded the data in widely separated subgroups to ensure maximum accuracy in compression and prediction tasks. Also, in light of the impairment caused by this behavior to the semantic structure of the space, we argued that a control over the latent space is mandatory during the learning of an embedding. Moreover, in seeking to pinpoint the semantic relationships between the embedded elements captured by the network, we observed that the meaning carried by a single note or chord is clearly context-dependent. Although the perception of a music excerpt remains determined by its context, wee argued that by increasing the granularity of our embeddings, the samples would carry a more general meaning and the semantic structure of the space will be reinforced.

Based on these hypotheses, we have defined a new approach detailed in Chapter 4. First, we have considered the core unit of embeddings to be an entire musical bar rather than a single event, ensuring that it could be described through generic features unrelated to the context in which it occurs. Then, we relied on another type of auto-encoders called VAE, which allows control over the latent space by pushing its data distribution to be close to a prior, in our case a Gaussian distribution with mean 0 and variance 1. We implemented an architecture similar to the recently proposed *MusicVAE*, which provides a hierarchical decoder that enhances the generation of complex data and forces the model to exclusively rely on the latent vectors to correctly decode the musical excerpts. Besides, a further significant element that has contributed to the success of this system is the use of a much more efficient input representation than the piano-roll. As this model has been initially targeting only monophonic data, this MIDI-like representation did not fit well with polyphonic sequences, which led us to develop our proposal. Thus, we have introduced the *signal-like* representation which aims to transform a piano-roll matrix into a small continuous waveform similar to an audio signal. By conducting an extended benchmark against the main representations of the literature, we have shown that, in the context of learning embedding spaces for polyphonic bars

with VAE, our representation improves learning stability and leads to a better reconstruction and KL divergence trade-off. Moreover, since analyzing the structure of such a space from a musical point of view is not trivial, we have presented a method relying on a principled synthetic dataset that allows assessing how certain music theory concepts have been encoded in the space. By these means, we have shown that the structure of the embeddings learned through our method reflects at least partially the semantics of the embedded elements. These results have led to a publication in the "2020 Joint Conference on AI Music Creativity" organized and hosted by the Royal Institute of Technology (KTH) in Stockholm, Sweden.

Beyond being an aid to all systems requiring symbolic representation as input, these VAE-learned spaces can be directly exploited as tools to foster musical creation by virtue of their semantic structures and their continuity properties. In Chapter 5, we have presented several applications to illustrate these abilities. First, we have proposed a composer identification tool by training a very simple RNN classifier that takes as input small sequences of 6 consecutive bars previously embedded by our pre-trained embedding model and attempts to classify them among 5 great composers of Western classical music. Although these results show a significant benefit in using embedding vectors as an input representation, the tool remains insufficiently accurate to be fully effective for automatic labeling or music composition analysis. However, by training separate networks to acknowledge the work of a single composer among the others, the performance achieved has become close to that expected from such tools. Then, we have introduced several methods intending to enhance the creativity of composers. Indeed, we have shown that we can extract attribute (i. e.fundamental feature) vectors from the meaningful structure of our spaces which can be used to purposefully shift the amount of the corresponding attribute exhibited in a bar. Additionally, we have seen how the interpolation between two latent vectors may engender the generation of a sequence of realistic bars that are smoothly evolving from one sample to the other. Therefore, we believe that our embedding model can be beneficial to the compositional process, either by providing quick and visual access to a range of meaningful bar alterations or by allowing musical rendering of geometrical or arithmetical patterns initiated by the composer.

6.2 FUTURE WORKS

In addition to the further development of all the methods presented here, we see that the work done during this thesis could be extended and serve as a seed for groundbreaking researches. Here, we propose some ideas for future works.

A first stream of research could be to address multimodal music analysis frameworks. Indeed, a musical piece can be described with different modalities (audio signal, score or perceptual effects), which carry different information but with a common semantic content. Hence, we are convinced that linking them could lead to a higher-level understanding of musical data and potentially powerful applications. The embedding approach seems to fit perfectly well for this task as different transformations can be applied to embed each modality in different spaces with identical dimensionality. The learning criterion for the multimodal model then simply becomes a minimization of the distance between semantically similar vectors from different modalities (Palatucci et al., 2009).

On the other hand, we need to be able to exploit efficiently our spaces to produce concrete tools for musical creativity. However, even if the dimensionality of the input space has been drastically reduced, it may remain too high for an intuitive exploration. As a further matter, some dimensions can be optimal non-linear combinations of the input, whereas other dimensions might be less relevant. Therefore, as future work, we could try to identify and discriminate information-carrying dimensions. Our first trail to handle this question is that we need to define measures derived from information geometry and topology to automatically evaluate the information content of various dimensions, such as musical information rate or topological variations (Bergomi, 2015).

6.3 OVERALL CONCLUSION

The goal of this thesis was to develop new representations of symbolic polyphonic music in an empirical manner through a machine learning framework.

We succeeded to build a very promising approach that aims to represent symbolic music objects in a space that carry semantic relationships between the elements. With many possible improvements, our model already shown very interesting results in representing polyphonic musical bars. Moreover, we proposed several applications based on such representations that could allow both to infer knowledge on musical concepts and to increase musical creativity.

BIBLIOGRAPHY

Albini, Giovanni and Samuele Antonini (2009). 'Hamiltonian cycles in the topological dual of the tonnetz.' In: *International Conference on Mathematics and Computation in Music*. Springer, pp. 1–10 (cit. on p. 11).

Allan, Moray and Christopher Williams (2005). 'Harmonising chorales by probabilistic inference.' In: *Advances in neural information processing systems*, pp. 25–32 (cit. on p. 52).

Apel, Willi (1961). *The notation of polyphonic music, 900-1600*. 38. Medieval Academy of Amer (cit. on p. 7).

Assayag, Gérard et al. (1999). 'Computer-assisted composition at IRCAM: From PatchWork to OpenMusic.' In: *Computer Music Journal* 23.3, pp. 59–72 (cit. on p. 8).

Aytar, Yusuf et al. (2016). 'Soundnet: Learning sound representations from unlabeled video.' In: *Advances in Neural Information Processing Systems*, pp. 892–900 (cit. on p. 3).

Bahdanau, Dzmitry et al. (2014). 'Neural machine translation by jointly learning to align and translate.' In: *arXiv preprint arXiv:1409.0473* (cit. on p. 30).

Bengio, Yoshua et al. (1994). 'Learning long-term dependencies with gradient descent is difficult.' In: *IEEE transactions on neural networks* 5.2, pp. 157–166 (cit. on p. 25).

Bengio, Yoshua et al. (2003). 'A neural probabilistic language model.' In: *Journal of machine learning research* 3.Feb, pp. 1137–1155 (cit. on pp. 33, 34).

Bergomi, Mattia Giuseppe (2015). 'Dynamical and topological tools for (modern) music analysis.' PhD thesis (cit. on p. 90).

Bigo, Louis and Moreno Andreatta (2014). 'A geometric model for the analysis of pop music.' In: *Sonus-special issue on modelling in musical analysis* 35.1, pp. 36–48 (cit. on p. 11).

— (2017). 'Towards Structural (Popular) Music Information Research.' In: *European Music Analysis Conference (EuroMAC 2017)* (cit. on p. 10).

Bigo, Louis et al. (2011). 'Building Topological Spaces for Musical Objects.' In: *MCM*. Springer, pp. 13–28 (cit. on p. 10).

Bigo, Louis et al. (2015). 'Representation of musical structures and processes in simplicial chord spaces.' In: *Computer Music Journal* 39.3, pp. 9–24 (cit. on p. 11).

Blei, David M et al. (2003). 'Latent dirichlet allocation.' In: *Journal of machine Learning research* 3.Jan, pp. 993–1022 (cit. on p. 33).

Blei, David M et al. (2017). 'Variational inference: A review for statisticians.' In: *Journal of the American statistical Association* 112.518, pp. 859–877 (cit. on p. 28).

Bosseur, Jean-Yves (2005). *Du son au signe: histoire de la notation musicale*. Editions Alternatives (cit. on p. 6).

Boulanger-Lewandowski, Nicolas et al. (2012). 'Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription.' In: *arXiv preprint arXiv:1206.6392* (cit. on pp. 42, 54).

Bowman, Samuel R et al. (2015). 'Generating sentences from a continuous space.' In: *arXiv preprint arXiv:1511.06349* (cit. on p. 43).

Bretan, Mason et al. (2017). 'Learning and Evaluating Musical Features with Deep Autoencoders.' In: *arXiv preprint arXiv:1706.04486* (cit. on p. 42).

Cauchy, Augustin (1847). 'Méthode générale pour la résolution des systemes d'équations simultanées.' In: *Comp. Rend. Sci. Paris* 25.1847, pp. 536–538 (cit. on p. 13).

Chen, Xi et al. (2016). 'Variational lossy autoencoder.' In: *arXiv preprint arXiv:1611.02731* (cit. on p. 65).

Collobert, Ronan and Jason Weston (2008). 'A unified architecture for natural language processing: Deep neural networks with multitask learning.' In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 160–167 (cit. on p. 33).

Crestel, Léopold and Philippe Esling (2016). 'Live Orchestral Piano, a system for real-time orchestral music generation.' In: *arXiv preprint arXiv:1609.01203* (cit. on p. 52).

Elman, Jeffrey L (1990). 'Finding structure in time.' In: *Cognitive science* 14.2, pp. 179–211 (cit. on p. 25).

Euler, Leonhard (1739). *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae*. ex typographia Academiae scientiarum (cit. on p. 10).

Firat, Orhan et al. (2016). 'Multi-way, multilingual neural machine translation with a shared attention mechanism.' In: *arXiv preprint arXiv:1601.01073* (cit. on p. 30).

Good, Michael et al. (2001). 'MusicXML: An internet-friendly format for sheet music.' In: *XML Conference and Expo*, pp. 03–04 (cit. on p. 8).

Gross, Richard (2015). *Psychology: The science of mind and behaviour 7th edition*. Hodder Education (cit. on p. 12).

Hadjeres, Gaëtan et al. (2017). 'Deepbach: a steerable model for bach chorales generation.' In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1362–1371 (cit. on pp. 8, 65).

Hahnloser, Richard HR et al. (2000). 'Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit.' In: *Nature* 405.6789, pp. 947–951 (cit. on p. 18).

Hawthorne, Curtis et al. (2018). 'Transformer-NADE for Piano Performances.' In: *NIPS Second Workshop on Machine Learning for Creativity and Design* (cit. on pp. 8, 9).

Hawthorne, Curtis et al. (2019). 'Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset.' In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1lYRjC9F7 (cit. on p. 75).

He, Kaiming et al. (2016). 'Deep residual learning for image recognition.' In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (cit. on pp. 23, 24).

Hinton, Geoffrey and Sam T Roweis (2002). 'Stochastic neighbor embedding.' In: *NIPS*. Vol. 15. Citeseer, pp. 833–840 (cit. on p. 39).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). 'Long short-term memory.' In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 25).

Hotelling, Harold (1933). 'Analysis of a complex of statistical variables into principal components.' In: *Journal of educational psychology* 24.6, p. 417 (cit. on p. 38).

Huang, Cheng-Zhi Anna et al. (2016). 'Chordripple: Recommending chords to help novice composers go beyond the ordinary.' In: *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, pp. 241–250 (cit. on p. 41).

Huang, Gao et al. (2016). 'Deep networks with stochastic depth.' In: *European conference on computer vision*. Springer, pp. 646–661 (cit. on p. 24).

Huang, Gao et al. (2017). 'Densely connected convolutional networks.' In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708 (cit. on p. 24).

Hubel, David H and Torsten N Wiesel (1962). 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex.' In: *The Journal of physiology* 160.1, pp. 106–154 (cit. on p. 22).

Karpathy, Andrej and Li Fei-Fei (2015). 'Deep visual-semantic alignments for generating image descriptions.' In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137 (cit. on p. 3).

Kingma, Diederik P and Jimmy Ba (2014). 'Adam: A method for stochastic optimization.' In: *arXiv preprint arXiv:1412.6980* (cit. on p. 53).

Kingma, Diederik P and Max Welling (2013). 'Auto-encoding variational bayes.' In: *arXiv preprint arXiv:1312.6114* (cit. on pp. 28, 29).

Kiros, Ryan et al. (2014). 'Multimodal Neural Language Models.' In: *Icml*. Vol. 14, pp. 595–603 (cit. on p. 3).

Kiros, Ryan et al. (2015). 'Skip-thought vectors.' In: *Advances in neural information processing systems*, pp. 3294–3302 (cit. on p. 33).

Klumpenhouwer, Henry James (1991). *A generalized model of voice-leading for atonal music*. Harvard University (cit. on p. 10).

Krizhevsky, Alex et al. (2012). 'Imagenet classification with deep convolutional neural networks.' In: *Advances in neural information processing systems*, pp. 1097–1105 (cit. on p. 22).

Landauer, Thomas K (2006). *Latent semantic analysis*. Wiley Online Library (cit. on p. 33).

Larochelle, Hugo and Iain Murray (2011). 'The neural autoregressive distribution estimator.' In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, pp. 29–37 (cit. on p. 41).

Larsson, Gustav et al. (2016). 'Fractalnet: Ultra-deep neural networks without residuals.' In: *arXiv preprint arXiv:1605.07648* (cit. on p. 24).

Leshno, Moshe et al. (1993). 'Multilayer feedforward networks with a nonpolynomial activation function can approximate any function.' In: *Neural networks* 6.6, pp. 861–867 (cit. on p. 19).

Lewin, David (1990). 'Klumpenhouwer networks and some isographies that involve them.' In: *Music Theory Spectrum* 12.1, pp. 83–120 (cit. on p. 10).

— (1994). 'A Tutorial on Klumpenhouwer Networks, Using the Chorale in Schoenberg's Opus 11, No. 2.' In: *Journal of Music Theory* 38.1, pp. 79–101 (cit. on p. 10).

Maaten, Laurens van der and Geoffrey Hinton (2008a). 'Visualizing data using t-SNE.' In: *Journal of Machine Learning Research* 9.Nov, pp. 2579–2605 (cit. on p. 70).

Maaten, Laurens Van der and Geoffrey Hinton (2008b). 'Visualizing data using t-SNE.' In: *Journal of machine learning research* 9.11 (cit. on p. 39).

Madjiheurem, Sephora et al. (2016). 'Chord2Vec: Learning musical chord embeddings.' In: *Proceedings of the Constructive Machine Learning Workshop at 30th Conference on Neural Information Processing Systems (NIPS'2016), Barcelona, Spain* (cit. on p. 41).

McCulloch, Warren S and Walter Pitts (1943). 'A logical calculus of the ideas immanent in nervous activity.' In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133 (cit. on p. 17).

Mersenne, Marin (1972). *Harmonicorum libri xii....* (Cit. on p. 10).

Mesnage, Marcel (1997). 'Entités formelles pour l'analyse musicale.' In: *GENEVOIS et ORLAREY*, pp. 93–109 (cit. on p. 10).

Mikolov, Tomas et al. (2013a). 'Distributed representations of words and phrases and their compositionality.' In: *Advances in neural information processing systems*, pp. 3111–3119 (cit. on p. 33).

Mikolov, Tomas et al. (2013b). 'Efficient estimation of word representations in vector space.' In: *arXiv preprint arXiv:1301.3781* (cit. on p. 33).

Mroueh, Youssef et al. (2015). 'Deep multimodal learning for audio-visual speech recognition.' In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pp. 2130–2134 (cit. on pp. 3, 33).

Noda, Kuniaki et al. (2015). 'Audio-visual speech recognition using deep learning.' In: *Applied Intelligence* 42.4, pp. 722–737 (cit. on p. 33).

Oord, Aaron van den et al. (2016). 'Conditional image generation with pixelcnn decoders.' In: *arXiv preprint arXiv:1606.05328* (cit. on p. 43).

Oore, Sageev et al. (2018). 'This time with feeling: Learning expressive musical performance.' In: *Neural Computing and Applications*, pp. 1–13 (cit. on pp. 8, 9).

Palangi, Hamid et al. (2016). 'Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval.' In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24.4, pp. 694–707 (cit. on p. 3).

Palatucci, Mark et al. (2009). 'Zero-shot learning with semantic output codes.' In: *Advances in neural information processing systems*, pp. 1410–1418 (cit. on p. 90).

Pascanu, Razvan et al. (2012). 'Understanding the exploding gradient problem.' In: *CoRR, abs/1211.5063* 2, p. 417 (cit. on p. 66).

Pearson, Karl (1901). 'LIII. On lines and planes of closest fit to systems of points in space.' In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572 (cit. on p. 38).

Pennington, Jeffrey et al. 'GloVe: Global Vectors for Word Representation.' In: URL: https://nlp.stanford.edu/projects/glove/ (cit. on p. 33).

Pennington, Jeffrey et al. (2014). 'Glove: Global Vectors for Word Representation.' In: *EMNLP*. Vol. 14, pp. 1532–1543 (cit. on pp. 33, 37).

Perle, George (1996). *Twelve-tone tonality*. Univ of California Press (cit. on p. 10).

Poliner, Graham E and Daniel PW Ellis (2006). 'A discriminative model for polyphonic piano transcription.' In: *EURASIP Journal on Advances in Signal Processing* 2007.1, p. 048317 (cit. on p. 52).

Raffel, Colin and Daniel PW Ellis (2016). 'Extracting Ground-Truth Information from MIDI Files: A MIDIfesto.' In: *ISMIR*, pp. 796–802 (cit. on p. 78).

Reed, Scott et al. (2016). 'Generative adversarial text to image synthesis.' In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 3 (cit. on p. 33).

Ren, Zhou et al. (2016). 'Joint Image-Text Representation by Gaussian Visual-Semantic Embedding.' In: *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, pp. 207–211 (cit. on p. 33).

Roberts, Adam et al. (2018). 'A hierarchical latent vector model for learning long-term structure in music.' In: *arXiv preprint arXiv:1803.05428* (cit. on pp. 43, 44, 62).

Rumelhart, David E et al. (1986). 'Learning representations by back-propagating errors.' In: *nature* 323.6088, pp. 533–536 (cit. on p. 20).

Rush, Alexander M et al. (2015). 'A neural attention model for abstractive sentence summarization.' In: *arXiv preprint arXiv:1509.00685* (cit. on p. 30).

Simonyan, Karen and Andrew Zisserman (2014a). 'Two-stream convolutional networks for action recognition in videos.' In: *Advances in neural information processing systems*, pp. 568–576 (cit. on p. 22).

— (2014b). 'Very deep convolutional networks for large-scale image recognition.' In: *arXiv preprint arXiv:1409.1556* (cit. on p. 22).

Smolensky, Paul (1986). *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. Colorado Univ at Boulder Dept of Computer Science (cit. on p. 42).

Srivastava, Rupesh Kumar et al. (2015). 'Training very deep networks.' In: *arXiv preprint arXiv:1507.06228* (cit. on p. 24).

Sutskever, Ilya et al. (2014). 'Sequence to sequence learning with neural networks.' In: *Advances in neural information processing systems*, pp. 3104–3112 (cit. on p. 41).

Tang, Duyu et al. (2014). 'Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification.' In: *ACL (1)*, pp. 1555–1565 (cit. on p. 3).

Typke, Rainer et al. (2004). 'Searching notated polyphonic music using transportation distances.' In: *Proceedings of the 12th annual ACM international conference on Multimedia*. ACM, pp. 128–135 (cit. on p. 10).

Uitdenbogerd, Alexandra and Justin Zobel (1999). 'Melodic matching techniques for large music databases.' In: *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM, pp. 57–66 (cit. on p. 10).

Ukkonen, Esko et al. (2003). 'Geometric algorithms for transposition invariant content-based music retrieval.' In: (cit. on p. 10).

Vaswani, Ashish et al. (2017). 'Attention is all you need.' In: *Advances in Neural Information Processing Systems*, pp. 5998–6008 (cit. on pp. 31, 47).

Vieru, Anatol (1995). *The Musical signification of Multipplication by 7. Diatonicity and Chromaticity* (cit. on p. 10).

Vinyals, Oriol et al. (2015). 'Show and tell: A neural image caption generator.' In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164 (cit. on p. 33).

Xu, Kelvin et al. (2015). 'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.' In: *ICML*. Vol. 14, pp. 77–81 (cit. on p. 30).