# Machine Learning Models for Multimodal Detection of Anomalous Behaviors

Valentin Durand de Gevigney

DOCTORAT BRETAGNE LOIRE MATHSTIC

Université Bretagne Sud

# THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD

ÉCOLE DOCTORALE N° 601
*Mathématiques et Sciences et Technologies de l'Information et de la Communication*
Spécialité : *Informatique*

Par

**Valentin DURAND de GEVIGNEY**

## Machine Learning Models for Multimodal Detection of Anomalous Behaviors

Modèles d'Apprentissage Automatique pour la Détection Multimodale de Comportements Anormaux

**Thèse présentée et soutenue à Lannion, le 10 décembre 2021**
**Unité de recherche : IRISA, UMR CNRS 6074**
**Thèse N° : 615**

### Rapporteurs avant soutenance :

| | |
|---|---|
| Mohamed DAOUDI | Professeur des Universités, IMT Lille Douai - CRIStAL |
| Germain FORESTIER | Professeur des Universités, Univ. Haute-Alsace - IRIMAS |

### Composition du Jury :

| | | |
|---|---|---|
| Président : | Mohamed DAOUDI | Professeur des Universités, IMT Lille Douai - CRIStAL |
| Examinateurs : | Mohamed DAOUDI | Professeur des Universités, IMT Lille Douai - CRIStAL |
| | Germain FORESTIER | Professeur des Universités, Univ. Haute-Alsace - IRIMAS |
| | Lauriane AUFRANT | Responsable de la mission TALN pour la Défense et sécurité, Inria - DGA |
| | Marie TAHON | Maître de Conférences, Univ. Le Mans - LIUM |
| Dir. de thèse : | Pierre-François MARTEAU | Professeur des Universités, Université de Bretagne Sud – IRISA |
| Co-dir. de thèse : | Damien LOLIVE | Maître de Conférences HDR, Université de Rennes 1 - IRISA |

### Invité(s) :

Encadrant Arnaud DELHAY-LORRAIN Maître de Conférences, Université de Rennes 1 - IRISA

# Acknowledgements / Remerciements

**English.** First and foremost, I would like to thank my father. He passed on his love for science to me, ultimately leading into the realization of the present thesis.

I also would like to thank all my teachers who helped me follow this path. Especially and in no particular order: Lecturer Olivier Christmann, Professor David Rudrauf, Associate Professor Arnaud Delhay-Lorrain, Professor Pierre-François Marteau and Associate Professor Damien Lolive.

My thanks also go to my close ones: Mina Goudour, my friends and the of rest my family. Their support was essential, not only during this thesis, but also to face the events created by covid-19 pandemic.

Furthermore, for their help in polishing this document, their comments, questions and improvements, I would like to thank the external members of the jury: Lauriane Aufrant, Marie Tahon, Mohamed Daoudi and Germain Forestier.

I also would like to thank the UBS (Université de Bretagne Sud) and the DGA (Direction Générale de l'Armement) for their financial support to this thesis.

And finally, the scientific community, which laid the foundations without which none of this work would have been possible.

**Français.** Tout d'abord, je tiens à remercier mon père. Il m'a transmis son amour de la science, ce qui a finalement mené à la réalisation de la thèse présentée dans ce document.

Je tiens également à remercier tous mes enseignants qui m'ont aidé à poursuivre cette voie. En particulier et dans un ordre arbitraire : Maître de Conférences Olivier Christmann, Professeur David Rudrauf, Maître de Conférences Arnaud Delhay-Lorrain, Professeur Pierre-François Marteau et Maître de Conférences Damien Lolive.

Mes remerciements vont également à mes proches : Mina Goudour, mes amis et le reste de ma famille. Leur support a été essentiel, non seulement pendant cette thèse, mais également pour affronter les évènements créés par la pandémie de covid-19.

Pour leur aide dans la finition de ce document, pour leurs commentaires, leurs questions

et leur améliorations, je remercie également les membres extérieurs du jury: Lauriane Aufrant, Marie Tahon, Mohamed Daoudi et Germain Forestier.

Je remercie également l'UBS (Université de Bretagne Sud) et la DGA (Direction Générale de l'Armement) pour leur soutien financier à cette thèse.

Et pour terminer, la communauté scientifique, qui a établi les fondations sans lesquelles aucune partie de ce travail n'aurait été possible.

# RÉSUMÉ EN FRANÇAIS

Cette section constitue le résumé substantiel en français du manuscrit de thèse, les chapitres restants étant écrits en anglais.

Les travaux effectués durant cette thèse concernent de développement de modèles qui ont pour but la détection de comportements humains anormaux. Ces modèles sont développés en considérant la mise à profit de plusieurs modalités — *e.g.* audio+vidéo — afin d'améliorer la détection d'anomalies.

Dans le chapitre 1 "Introduction", nous commencerons par faire un inventaire détaillé — mais non exhaustif — des nombreuses motivations derrière nos travaux. Ensuite, nous définirons plus précisément le problème de la détection d'anomalie. En particulier, nous nous arrêterons sur la notion d'anomalie qui est parfois contre-intuitive. Nous restreindrons ensuite à la détection de comportements anormaux, en prenant soin d'appuyer sur la nature ambiguë de la tâche. Après cela, nous aborderons l'aspect multimodal de nos tâches en détaillant une à une les différentes modalités qui nous intéressent. Enfin, pour terminer ce premier chapitre, nous parlerons brièvement des travaux de thèse de Cédric Fayet, dont nos travaux sont une suite directe.

Dans le chapitre 2 "État de l'art", nous donnerons premièrement une vue d'ensemble sur les méthodes employées dans les domaines les plus proches du notre. Nous expliciterons également quelles méthodes nous avons choisies d'employer pour nos travaux. Deuxièmement, nous ferons un inventaire de jeux de données liés à notre tâche et qui peuvent nous servir d'une manière ou d'une autre. Pour finir ce chapitre, nous parlerons des différentes limitations de l'état de l'art que nous avons identifiées et grâce auxquelles nous avons pu définir la trajectoire de nos travaux.

Dans le chapitre 3 "Méthodologie", nous présenterons trois ensembles de méthodes que nous utiliserons dans nos contributions. Premièrement, nous parlerons des différents paradigmes d'apprentissage à notre disposition et de ceux que nous privilégierons. En deuxième, nous détaillerons plusieurs classes de modèles d'apprentissage profond qui nous intéressent pour la détection d'anomalies, de manière directe ou indirecte. Enfin, nous aborderons différentes méthodes que nous pouvons utiliser afin d'estimer les performances d'un système de détection d'anomalies, ainsi que l'importance de ces différentes méthodes.

Dans le chapitre 4 "Contributions principales", nous détaillerons nos résultats principaux qui sont composés de trois modèles pour la détection d'anomalies. Chaque modèle recevra une attention particulière afin de détailler la motivation derrière leur développement, la manière dont ils sont construits, la manière dont ils sont utilisés pour détecter des anomalies, les résultats que ces modèles obtiennent et enfin une discussion sur ces modèles afin de voir leurs forces, leurs faiblesses et leurs perspectives.

Dans le chapitre 5 "Autres contributions", nous aborderons des méthodes supplémentaires, dont le but premier n'est pas la détection d'anomalies mais de fournir de nouveaux outils à cette dernière. Enfin, nous parlerons de nos trouvailles sur le sujet de l'apprentissage de représentations, un outil qui sera essentiel à l'avenir pour modéliser correctement les comportements humains, et donc détecter des comportements humains anormaux.

Dans le chapitre 6 "Discussion", nous reviendrons d'abord sur la nature contre-intuitive de la notion d'anomalie. Ensuite, nous parlerons du fait que l'intérêt de plusieurs modalités est sous-évalué et que la pluralité des modalités est parfois tout simplement nécessaire. Après cela, nous reviendrons sur un de nos résultats qui concerne les a priori inscrits dans les modèles et de leur relation à la détection d'anomalies. Par la suite, nous parlerons du fait que les comportements humains peuvent être considérés comme des mécanismes temporels invisibles mais pourtant bien présents. Ensuite, nous reviendrons sur nos résultats concernant l'apprentissage de représentations, du travail qu'il reste à faire dessus et sur le fait que nous avons déjà une bonne partie des outils nécessaires à cet apprentissage. Pour terminer, nous reviendrons également sur les difficultés posées par la nature changeante du monde réel et ce que nos résultats semblent indiquer pour aborder ces difficultés.

Dans le chapitre final "Travaux futurs", nous ouvrirons des perspectives qui découlent de l'ensemble de nos travaux — des problématiques posées, de nos résultats et de notre discussion — à commencer par les travaux futurs nécessaires sur l'apprentissage de représentations (de comportement humains en particulier). Ensuite, nous parlerons de travaux futurs que nous estimons essentiels à réaliser sur les jeux de données, en particulier pour la détection multimodale de comportements humains anormaux. Nous terminerons avec les travaux futurs que nous anticipons sur l'apprentissage actif d'anomalies afin que les modèles soient capables de suivre les changements perpétuels du monde réel.

# Introduction

Les humains affichent en permanence des comportements qui sont la réflection de leur état interne et de leur cognition interne. Les conséquences que ces comportements peuvent avoir — sur les individus eux-mêmes ou leur entourage — peuvent appeler à une réaction qui est parfois même vitale.

En particulier, il est possible de définir les comportements normaux comme ceux auxquels on ne souhaite pas réagir et par extension, les comportements anormaux deviennent ceux qui demandent une réaction.

À cause de la quantité totale de situations où ces comportements anormaux sont présents, il est souhaitable de d'automatiser leur détection. Si ces situations sont nombreuses au total, elles sont également très diverses.

## Cas d'application

**Sécurité.** La sécurité est généralement le premier cadre d'application auquel on pense : on préfère prévenir les incidents comme les agressions, plutôt que devoir les subir.

**Pilotage.** Le pilotage de n'importe quel engin (*e.g.* avion, construction, *etc.*) est un autre cadre d'application possible pour la détection de comportements anormaux à partir du moment où un comportement normal — comme du stress — induit un danger.

**Santé mentale.** Plus subtile, la santé mentale est également un cas d'application possible très prometteur. Il est préférable d'identifier une mauvaise santé mentale — comme un état dépressif — le plus tôt possible afin d'en minimiser les conséquences qui peuvent être graves.

**Handicaps invisibles.** Aujourd'hui, le diagnostic de handicaps invisibles est toujours difficile à cause de la nature invisible de ces handicaps. Cependant, de par le fait qu'ils sont des handicaps, ils affectent directement ou indirectement le comportement. Les détecter plus facilement et plus tôt permettra une meilleure prise en charge de ces handicaps qui le sont encore très insuffisamment de nos jours. Parmi ces handicaps invisibles, on compte entre autres les troubles psychiques (*e.g.* le trouble anxieux, le trouble de stress post-traumatique ou encore le trouble du déficit d'attention, ...).

**Flux de réseaux informatiques.** Une partie du réseau informatique est une conséquence directe de l'activité humaine, et donc de comportements humains. Pour de multiples raisons, généralement similaires à celles que nous avons déjà listées, il peut être souhaitable d'être en mesure de détecter une activité humaine anormale.

**Vote en ligne / électronique.** Probablement voué à être étendu à l'avenir, le vote électronique suscite beaucoup d'inquiétudes dont une partie est liée à des comportements humains. Nous ne parlons pas seulement ici de votes à des élections, mais de toute forme de vote en ligne, comme des systèmes collaboratifs. Si une personne ne semble pas avoir son comportement normal, cela peut-être un indice que cette personne ne vote pas selon sa volonté (ou que c'est pas une personne du tout).

En résumé, la détection de comportements humains anormaux peut servir dans les domaines suivants : La sécurité, la santé mentale, le diagnostic de handicaps invisibles, les réseaux informatiques, le vote en ligne, *etc.*

## La détection d'anomalies

Tournons nous d'abord vers les définitions couramment données à la notion d'anomalies (traduites depuis l'anglais). Chandola, Banerjee, and Kumar 2009 définissent les anomalies comme : *[...] des motifs dans les données qui ne se conforment pas à une notion bien définie de normalité.*

De son côté, D. M. Hawkins 1980 donne une définition pour les *outliers*, un concept similaire : *Un outlier est défini comme une observation qui diffère tellement des autres observations qu'on en vient à suspecter qu'elle a été générée par un mécanisme différent.*

Dans les deux cas, il est important de noter que l'on ne définit jamais directement les anomalies. À la place, on définit ce qui est normal. Ensuite, par opposition, tout ce qui n'est pas normal est donc anormal.

Cette distinction est importante à faire parce que si les caractéristiques des observations normales peuvent être énumérées, ce n'est jamais le cas pour les anomalies. Les anomalies sont — par définition — inattendues, imprévisibles, impossibles à modéliser.

Il nous faut donc d'abord un ensemble de règles qui définissent ce qui est normal. Nous ferons référence à cet ensemble de règles en utilisant le terme de "contexte".

Par conséquent, ce qui est normal dans un contexte ne le sera pas forcément dans un autre contexte. Sans contexte, la notion d'anomalie n'a pas de sens.

### Comportements humains anormaux

La notion de comportement humain est en elle-même très floue. Cela s'observe notamment par le fait qu'il est très aisé de confondre plusieurs comportements (*e.g.* confondre de l'impatience pour de l'arrogance, de l'ironie pour du sarcasme, *etc.*)

Par conséquent, les utilisateurs souhaitant un jour déployer un système de détection de comportements anormaux devront être au courant de l'aspect vague induit par la notion de comportement humain. Et ce afin que ces utilisateurs puissent préciser au mieux leur pensée et donc le contexte sur lequel repose entièrement la notion d'anomalie.

### Détection multimodale d'anomalies

Les comportements humains peuvent être observés à travers une grande diversité de modalités. Nous en listons une partie significative ici : La vidéo (et ses dérivés comme l'extraction du flux optique qui se concentre sur le mouvement, ou encore les points-clés comme ceux qui identifient des points précis du visage ou du corps); Le son (et ses dérivés comme le spectre sonore, particulièrement adapté à la parole); Le texte (*e.g.* sous forme de caractères); Les capteurs de signaux corporels (comme le rythme cardiaque, l'activité électrique du cerveau, la sudation, *etc.*); Les flux réseau; Les votes électroniques.

### Détection multimodale d'anomalies dans le discours à travers la parole et les expressions faciales

La présente thèse est la continuité de la thèse de Fayet 2018. Dans ses travaux, Fayet s'est concentré sur le cadre du discours et sur l'expressivité de la voix et du visage, puisqu'elle contient des informations sur l'état émotionnel et l'état d'esprit d'une personne. De manière plus importante pour les travaux de la présente thèse est la création du jeu de données Emo&ly par Fayet et al. 2018. Il s'agit, à notre connaissance, du premier et seul jeu de données pour la détection multimodales de comportements humains anormaux. Nous entrerons dans plus de détails lorsque nous parlerons de tous les jeux de données.

## État de l'art

Nous parlons maintenant de l'état de l'art, des différentes méthodes utilisées pour la détection d'anomalies — sans pour autant toutes les lister —, des différents jeux de

données liés à notre tâche et finalement des limitations de cet état de l'art, nous permettant de définir l'objectif de nos travaux.

## Méthodes de l'état de l'art

Les méthodes de l'état de l'art sont très nombreuses, mais il est possible de les regrouper en deux catégories : celles qui sont plus adaptées pour traiter des données de faible dimensionnalité — soit celles qui ont peu de caractéristiques — en faibles quantité et les méthodes qui sont plus adaptées pour l'inverse, à savoir traiter des données de haute dimensionnalité — comme la vidéo ou le son brut — en grandes quantités.

La première catégorie regroupe des méthodes bien étudiées et généralement robustes comme les One-Class Support Vector Machines (OC-SVMs), les Isolation Forests (iForests) et la méthode Local Outlier Factor (LOF). Cependant, ces méthodes souffrent du phénomène communément appelé le "Fléau de la dimensionnalité", un phénomène où les espaces à haute dimensionnalité deviennent extrêmement éparses, réduisant la capacité discriminative des mesures de distance dans ces espaces.

Ici, la seconde catégorie fait référence à l'apprentissage profond, une approche utilisant des réseaux de neurones et qui est capable de s'attaquer à des espaces à haute dimensionnalité. Pour la détection d'anomalies, un très grand nombre de modèles existe. La plus grande famille de méthode d'apprentissage profond pour la détection d'anomalies est la famille des autoencoders.

Pour faire simple, les autoencodeurs apprennent à compresser les données, puis à les décompresser tout en limitant les dégâts qu'ils font à ces données. On dit qu'ils apprennent à minimiser l'erreur de reconstruction. En pratique, les autoencodeurs sont très utilisés pour la détection d'anomalies car l'erreur de reconstruction est généralement un bon indicateur de normalité (une plus faible erreur indiquant une plus grande normalité).

Cependant, il existe de nombreuses autre familles, comme celle des modèles adversariaux. Cette famille de modèle est généralement constituée de deux sous-modèles, que l'on peut voir comme un fraudeur et un expert. Le fraudeur apprend à berner l'expert qui lui apprend à distinguer le vrai du faux. Les deux s'améliorent alors sans cesse grâce à l'amélioration de l'autre. Dans le cadre de la détection d'anomalies, on demandera alors à l'expert — après entraînement — ce qu'il pense de nos nouvelles données. Plus l'expert pensera que la donnée est fausse, plus on considérera qu'elle est anormale.

Évidemment, aucune famille de méthode de détection d'anomalie n'est parfaite et la recherche sur le sujet continue.

## Jeux de données

Nous avons utilisé deux catégories de jeux de données tout au long de cette thèse. La première catégorie est celle des jeux de données de détection de comportement anormaux. La seconde est celle des jeux de données qui ne servent pas directement à la détection d'anomalies mais peuvent nous y aider (typiquement, des jeux de données multi-modaux). Dans la première catégorie, à l'exception de Emo&ly, aucun des jeux de données n'est multi-modal, d'où le besoin de jeux de données multi-modaux supplémentaires, même s'ils ne servent pas à la détection d'anomalies.

### Jeux de données pour la détection de comportements anormaux

**Emo&ly.** Dans la première catégorie, nous retrouvons tout d'abord le jeu de données Emo&ly. Il s'agit de personnes filmées face-caméra et qui lisent des passages d'un conte dans trois situations différentes. Dans la première, rien ne perturbe la lecture. Dans la seconde, l'expérimentateur perturbe activement les lecteurs. Dans la troisième, il est demandé aux lecteurs de jouer la comédie pendant qu'ils lisent. Le jeu de données a par la suite été partiellement étiqueté, les réactions étant étiquetées comme anormales lorsque les annotateurs les considéraient inattendues.

**Surveillance vidéo.** Ensuite, nous retrouvons six jeux de données de surveillance vidéo, à savoir : 1) UCSD Pedestrian 2) Subway dataset (à obtenir auprès de email.amitadam@gmail.com) 3) CUHK Avenue 4) ShanghaiTech campus, 5) ITTB-Corridor et 6) UMN crowd activity.

**Surveillance audio.** Après ça, nous retrouvons un jeu de données de surveillance audio, à savoir le jeu de données MIVIA road audio events. Enfin, toujours dans la première catégorie, nous retrouvons le jeu de données Kitsune (qui peut être obtenu sur le site de l'UCI ou sur Kaggle).

### Jeux de données auxiliaires

**Audioset.** Dans la seconde catégorie, nous retrouvons tout d'abord le jeu de données Audioset (Gemmeke et al. 2017). Ce jeu de données est constitué de courtes vidéos venant de Youtube, pour un total de 5500 heures, dont 3 heures identifiées comme ayant de la parole humaine. Cela nous donne un jeu de données multimodal — audiovisuel plus

précisément — ce qui nous intéresse pour la modélisation multimodale des comportements humains.

**Kinetics.** Il existe également le jeu de données Kinetics (Smaira et al. 2020), un jeu de données pour la reconnaissance d'action en vidéo. Il s'agit également d'une collection de vidéos de Youtube mais contrairement à Audioset, toutes les vidéos n'ont pas de son.

**Tournesol.** Enfin, le jeu de données du projet Tournesol (Hoang 2021) est un jeu de données potentiel pour l'apprentissage sur des données de vote en ligne, puisque Tournesol est un projet de vote collaboratif à propos des vidéos en ligne, afin d'estimer leur importance d'un point de vue moral (de manière individuelles, pas dans leur globalité).

## Limitations de l'état de l'art

Nous avons identifié trois types de limitations dans l'état de l'art : 1) celles qui sont liées aux jeux de données, 2) celles qui sont liées à la méthodologie communément utilisée dans la littérature et 3) celles qui sont liées aux modèles développés dans l'état de l'art.

### Limitations dans les jeux de données

Tout d'abord, les jeux de données manquent généralement de données, cela réduit la difficulté qu'ils posent et donc la confiance en la robustesse des méthodes qui sont évaluées dessus.

Ensuite, les jeux de données ont parfois une définition floue — et/ou directe — de leur anomalies, ce qui rend l'évaluation plus subjective.

Enfin, ces jeux de données n'ont pas de jeu de validation, servant à s'assurer que le modèles ne sur-apprend pas. Étant donné la petite taille des jeux de données, un tel sur-apprentissage est plus que probable lorsque les marges d'améliorations sont faibles, comme on l'observe dans l'état de l'art sur la détection d'anomalies.

### Limitations dans la méthodologie

Premièrement, le faible nombre de jeux de données utilisés pour évaluer une méthode. Ce nombre tourne généralement autour de 2. Étant donné de nouveau la petite taille des jeux de données, ce faible nombre de jeux de données utiliser nous fait douter de la

robustesse des méthodes développées dans l'état de l'art, qui pourrait être inférieure à celle annoncée.

Deuxièmement, très peu d'auteurs fournissent les moyens de reproduire leurs travaux, ce qui ralentit l'expérimentation, la possibilité d'utiliser un jour leur méthode dans le monde réel et également la confiance en les méthodes ainsi développées.

Pour finir, les auteurs se contentent généralement d'analyses superficielles de leurs méthodes, comparant leurs méthodes en utilisant une unique valeur qui ne reflète pas la complexité de la tâche et ne cherchant pas à prouver la robustesse de leur méthode sous différentes conditions.

### Limitations des méthodes de l'état de l'art

Ces limitations, qui sont liées aux méthodes de l'état de l'art, sont celles qui forment de nouvelles problématiques et nous invitent au développement de nouveaux modèles.

Tout d'abord, nous notons l'absence de compensation pour le "concept drift", une notion qui encapsule le fait que le monde change. En effet, ce qui est vrai aujourd'hui ne le sera pas forcément demain. Nous trouvons cet élément particulièrement important lorsque l'on parle de comportements humains, un modèle figé dans le temps serait incapable de prendre en compte le progrès social sans être totalement reconstruit.

Ensuite, les méthodes ne cherchent pas à détecter la présence d'être humains. Ainsi, en utilisant les méthodes actuelles de l'état de l'art, il est tout à fait envisageable que le système détecte un comportement humain anormal sans qu'aucun humain ne soit présent.

Enfin, s'il existe de la littérature sur la détection de comportement anormal et de la littérature sur la détection multimodale d'anomalie, il n'existe — à notre connaissance — aucune littérature sur la détection multimodale de comportements humains anormaux (à l'exception des travaux de (Fayet 2018)). Dans ce domaine, tout est encore à faire.

# Méthodologie

Dans ce chapitre, nous détaillons trois ensembles de méthodes que nous utiliserons dans le chapitre suivant qui contient nos contributions.

## Paradigmes d'apprentissage

Nous commençons avec les paradigmes d'apprentissage. Ceux-ci ont un impact important sur, non seulement les modèles que nous pouvons entraîner, mais également sur les données dont nous avons besoin. Ces paradigmes d'apprentissage définissent quelles données un modèle utilise pour s'entraîner et en partie comment elles ont utilisées à cette fin.

### Apprentissage supervisé

L'apprentissage supervisé ressemble en tout points à un cahier d'exercices que les jeunes élèvent utilisent. Le cahier contient un ensemble de problèmes et la réponse pour chaque problème est disponible à la fin du cahier. L'apprentissage supervisé tire son nom du fait que l'élève (ou le modèle) est supervisé par la connaissance des réponses aux problèmes. Ainsi, l'élève (ou le modèle) regarde l'exercice, tente d'y répondre, compare sa réponse à celle dans le livre et en tire une leçon.

Typiquement, en classification d'images, on présente au modèle une image et le modèle répond avec une classe (*e.g.* chat, chien, cheval, *etc.*). Pour chaque image dans notre base de données, on connaît également sa classe, ce qui nous permet de calculer l'erreur du modèle.

Cependant, en détection d'anomalies, les anomalies ne présentent pas de caractéristiques particulières et on évitera autant que possible l'apprentissage supervisé pour cette raison. En effet, le modèle aura tendance à associer la classe "anomalie" à des caractéristiques particulières — et de même pour la classe "normal" — et lorsqu'il sera présenté avec une anomalie avec des caractéristiques différentes, le modèle ne saura pas dans quelle classe ranger l'échantillon.

Le défaut majeur de ce paradigme — pas seulement pour la détection d'anomalies — est le besoin d'associer à chaque échantillon une classe (ou une autre donnée). Le coût d'étiquetage — en temps, personnel et argent — grimpe alors rapidement avec les besoins modernes en quantité de données.

### Apprentissage non supervisé

L'apprentissage non supervisé, comme son nom l'indique, consiste à apprendre sans connaître la "réponse" à un problème. Ce paradigme permet entre autre de se passer complètement de la phase d'étiquetage.

En revanche, celui-ci force à trouver une méthode qui peut se passer d'étiquettes. Un exemple très utilisé, notamment en détection d'anomalies, est l'autoencodeur qui — pour faire simple — compresse et décompresse une donnée tout en apprenant à minimiser l'erreur entre la donnée d'entrée et sa reconstruction.

Dans la littérature, ce paradigme est assez rarement utilisé pour la détection d'anomalies. En effet, dû à l'absence totale d'étiquettes, le jeu de données contient à la fois des données normales et des anomalies (en très faible quantité).

#### Apprentissage semi-supervisé

C'est ce qui motive l'utilisation de l'apprentissage semi-supervisé en détection d'anomalies. Afin d'éviter que le modèle n'apprenne des données anormales, le jeu de données servant à l'entraînement est étiqueté afin d'en retirer les données anormales.

À noter que dans un cadre plus large, l'apprentissage semi-supervisé est le fait d'apprendre à partir de données dont seule une faible quantité a été étiquetée.

#### Apprentissage auto-supervisé

L'apprentissage auto-supervisé peut être vu comme une forme d'apprentissage non supervisé. Ce paradigme d'apprentissage vise à exploiter l'ordre intrinsèquement présent dans les données.

Par exemple, on peut créer deux vues différentes d'une même image — grâce à des opérations comme un rognage, un changement de teinte, une rotation, *etc.*— et créer une tâche qui exploite ces deux vues. Avec une vidéo qui a une bande audio, on peut également chercher à exploiter l'alignement temporel entre les deux bandes et leur relation d'un point de vue sémantique.

## Classes de modèles

Nous définissons maintenant cinq classes de modèles que nous utiliserons dans nos contributions. Ces classes définissent à la fois la structure des modèles et la manière dont ils sont entraînés. Nous noterons également que ces classes ne sont pas exclusives et peuvent donc être combinées entre elles.

### Autoencodeurs

Les autoencodeurs sont des modèles composés de deux parties : un encodeur et un décodeur. L'encodeur compresse les données d'entrée et le décodeur les décompresse. Puisqu'il s'agit d'un compression avec perte, l'autoencodeur est entraîné à minimiser cette perte, c'est à dire à minimiser la différence entre les données d'entrée et les données de sortie (aussi appelées "reconstructions").

Les autoencodeurs connaissent aussi quelques variantes. Il existe des autoencodeurs qui disposent de deux décodeurs par exemple, que nous appelons "Prédicteurs". Ces prédicteurs fonctionnent sur des séquences (*e.g.* de la vidéo). Le premier décodeur a le même objectif que d'habitude mais le second décodeur, lui, cherche à prédire la suite de la séquence/vidéo à partir de la compression de la vidéo présente.

Les autoencodeurs dits variationnels n'encodent pas directement les données vers une représentation compressée. À la place, ceux-ci produisent une distribution — typiquement normale — et la représentation compressée est ensuite obtenue en échantillonnant cette distribution. Le décodeur alors opère comme d'habitude.

Quant à lui, le "U-Net" introduit des connections à différentes profondeurs de l'autoencodeur. Ainsi, la sortie de l'encodeur à la profondeur choisie est directement connectée à l'entrée du décodeur à la même profondeur, ce qui inclut une connexion directe entre l'entrée et la sortie du U-Net. À chaque connexion moins profonde, les données sont moins compressées (et pas du tout pour la connexion la moins profonde). Il est important de noter que les U-Net ne sont pas simplement utilisés pour autoencoder, mais pour produire une sortie différente. Typiquement, les U-Net ont été développés pour segmenter des images d'imagerie médicales.

### Modèles adversariaux

Les modèles adversariaux décrivent des modèles où plusieurs sous-modèles sont mis en compétition. L'exemple le plus connu est celui des Generative Adversarial Networks (GANs) que nous décrivions plus haut, où un fraudeur et un expert sont mis en compétition pour générer des données réalistes.

Il existe aussi de nombreuses variantes de modèles adversariaux, dont une version qui remplace le générateur/fraudeur par un autoencodeur variationnel. Typiquement, les modèles adversariaux sont combinés à d'autres modèles afin d'augmenter le réalisme apparent des sorties de ces modèles.

### Modèles autorégressifs

Les modèles autorégressifs observent une séquence de données et produisent le prochain élément de cette séquence. Cet élément est ensuite ajouté à la séquence et l'opération recommence ainsi de suite.

Les transformers sont un exemple de modèle autorégressif qui ont fortement impacté la littérature scientifique sur le domaine du traitement automatique du langage naturel. En particulier, ces transformers utilisent un mécanisme à chaque étape qui leur permet de concentrer leur attention sur certains éléments particuliers de la séquence.

### Modèles basés sur l'énergie

Les modèles basés sur l'énergie reçoivent plusieurs données d'entrée et déterminent la compatibilité entre ces différentes données. Là où un modèle classique de classification d'images recevrait une image et produirai la classe de cette image en sortie, un modèle d'énergie reçoit à la fois l'image et l'étiquette et produit une valeur d'énergie en sortie. Le modèle est alors entraîné à minimiser cette valeur d'énergie lorsque l'image et l'étiquette correspondent et à maximiser cette valeur lorsqu'elles ne le sont pas.

Si cela n'est pas toujours évident de prime abord, une très grande quantité de problèmes peuvent être reformulés de manière à donner une modèle de classification — comme la classification —, ce qui ouvre de nouvelles perspectives.

### Modèles de méta-apprentissage

Là où les modèles classiques apprennent à accomplir une tâche sur un jeu de données, les modèles de méta-apprentissage apprennent à apprendre à accomplir des tâches. Ces modèles peuvent être longuement entraînés comme les modèles classiques. Une fois fait, ils auront appris à apprendre pourront donc se spécialiser sur de nouvelles données avec très peu d'exemples.

Une formulation de modèles de méta-apprentissage qui rencontre du succès dans la littérature consiste à créer un modèle qui dispose d'au moins les deux parties suivantes : un contrôleur et une mémoire externe. L'objectif du contrôleur est d'apprendre à résoudre des problèmes en lisant dans la mémoire et — plus important — à écrire dedans. Lorsque la tâche change, il suffit de vider la mémoire et de laisser le contrôleur la remplir en observant la nouvelle tâche. Cela permet entre autre de créer des modèles qui s'adaptent particulièrement bien à de nouveaux environnements ou des environnements changeants.

## Évaluation d'un système de détection d'anomalies

Une fois une méthode de détection d'anomalies développée, il est nécessaire d'évaluer ses performances pour vérifier que celle-ci est fonctionnelle et de la comparer aux méthodes précédentes. À l'instar de nombreux autres domaines, il n'est pas possible de dire qu'une méthode est supérieure à une autre en utilisant une seule métrique. À la place, nous disposons de plusieurs métriques qui mesurent différents aspects des performances d'une méthode.

Dans le cadre de la détection d'anomalie, le choix de la métrique la plus pertinente ne doit pas être fait par la personne qui développe le système mais par celle qui est susceptible de s'en servir. En effet, ce choix dépend très fortement du cadre d'application. Dans certaines situations, il est critique d'avoir une méthode qui détecte toutes les anomalies alors que dans d'autres, on préférera avoir une méthode qui est "sûre" d'elle lorsqu'elle détecte une anomalie.

Ce sont ces raisons qui nous poussent à évaluer nos méthodes de détection d'anomalies en utilisant plusieurs métriques. Cependant, dans la littérature, il est peu courant de voir plus d'une ou deux métriques être utilisées. Par conséquent, il ne nous est pas toujours possible de comparer nos méthodes sur toutes ces métriques. En revanche, en raison de l'importance d'avoir plusieurs métriques, nous nous en servons tout de même afin de notamment proposer une base de comparaison pour des travaux futurs.

### Sensibilité et Spécificité

La sensibilité — aussi connue sous le nom de Taux de Vrais Positifs (TVP, ou TPR en anglais) — mesure la fraction de toutes les anomalies connues qui ont été identifiée comme des anomalies. Sa contrepartie est le Taux de Faux Positifs (TFP, ou FPR en anglais) et désigne la fraction de fausses alarmes.

La spécificité — aussi connue sous le nom de Taux de Vrais Négatifs (TVN, ou TNR en anglais) — mesure la fraction de tous les échantillons normaux qui ont été identifiés comme normaux. Sa contrepartie est le Taux de Faux Négatives (TFN, ou FNR en anglais) et désigne la fraction d'anomalies qui ont été manquées.

Il est important de noter que les méthodes de détection d'anomalies ne produisent typiquement pas directement un simple booléen normal/anormal. À la place, celles-ci produisent un score d'anomalie, représenté par un simple scalaire. Ensuite, ce scalaire est comparé à un seuil déterminé et s'il est supérieur à ce seuil, alors l'échantillon est

considérer comme anormal.

La définition de ce seuil est normalement laissée à l'utilisateur potentiel de la méthode, qui le choisira en fonction de son cahier des charges. En conséquence, les méthodes de détection d'anomalies sont généralement testées sur un ensemble de seuils et les métriques — comme la sensibilité et la spécificité — sont alors tracées en fonction de ce seuil.

### Courbe ROC

La courbe ROC — pour *Receiver Operating Curve* en anglais — est obtenue en affichant le taux de vrais positifs (TVP) en fonction du taux de faux positifs (TFP). Les deux taux sont calculés sur les mêmes seuil. C'est à dire que, pour chaque point de la courbe de coordonnées $(TFP_i, TVP_i)$, ces coordonnées $TFP_i$ et $TVP_i$ sont obtenues pour un même seuil $s_i$.

Afin de comparer deux courbes ROC, on utilise généralement les aires sous ces courbes afin de les résumer en une seule valeur, bien que l'on y perde un peu d'information. Cela est motivé par le fait que — en général — plus l'aire sous une courbe ROC est grande, meilleure est la méthode. En effet, une aire plus grand signifie généralement qu'à un seuil donné, le taux de vrais positifs augmente et le taux de faux positifs diminue.

Il est important de noter que l'aire sous la courbe ROC est de loin l'outil de mesure de performance le plus utilisé dans la littérature sur la détection d'anomalies. Dans la très grande majorité des cas, si une seule ou deux métriques sont rapportées dans une publication, l'aire sous la courbe ROC en fera partie.

Une autre métrique peut être tirée de la courbe ROC : le taux d'erreurs égales, plus connu sous le nom de *Equal Error Rate* (EER) en anglais. L'EER indique le taux de faux positifs lorsque celui-ci devient égal au taux de faux négatifs. Par conséquent, contrairement à l'aire sous la courbe ROC, une valeur plus basse d'EER indique de meilleures performances.

### Précision et Rappel

La précision est définie comme la fraction des échantillons qu'un système considère comme anormale qui sont réellement anormaux, c'est à dire : $\frac{TVP}{TVP+TFP}$. Cette métrique permet de répondre à la question : "En sachant que le système indique qu'un échantillon est anormal, quelle est la probabilité que cet échantillon soit vraiment anormal ?" Cette métrique est particulièrement importante lorsque les anomalies sont très rares.

Le rappel, quant à lui, est simplement un autre nom pour la sensibilité (ou le taux de vrais positifs).

Il est alors possible de tracer la précision en fonction du rappel, à l'instar de la courbe ROC qui trace le taux de vrais positifs en fonction du taux de faux positifs. Bien que plus rarement utilisée dans la littérature, la courbe de Précision-Rappel apporte des informations importantes. Notamment, la courbe de Précision-Rappel donne une vision plus réaliste des performances de la méthode lorsque les anomalies sont particulièrement rares, ce qui correspond bien mieux à une situation réelle.

De manière similaire à la courbe ROC, la courbe de Précision-Rappel peut-être résumée grâce à l'aire sous la courbe, une plus grande valeur indiquant encore de meilleures performances en moyenne.

# Contributions principales

Durant cette thèse, nous avons travaillé sur plusieurs aspects complémentaires liés directement ou indirectement à notre problématique. Nous aborderons d'abord les résultats principaux, c'est à dire aux modèles de détection de comportements anormaux. Ensuite, nous présenterons des modèles complémentaires et enfin, nous présenterons nos trouvailles sur l'apprentissage de représentations et son intérêt dans notre contexte.

## L'IAE

**Motivation.** L'objectif de cette méthode est de modéliser les comportements comme des mécanismes temporels. Ces fonctions ne prennent qu'un seul paramètre : le temps $t$ (soit la progression dans l'action).

**Méthode.** L'IAE (autoencodeur interpolant) est une méthode à base d'autoencodeur (compression-décompression) qui apprend à prédire le présent à partir du passé et du futur en interpolant les codes issus de l'encodeur. Pour faire simple, on se retrouve avec l'équation suivante :

$$code(t) = code_{\text{passé}} * (1 - t) + code_{\text{futur}} * t \qquad \text{avec } t \in [0, 1]$$

**Détection d'anomalie.** Pour détecter les anomalies, il y a deux options : utiliser l'IAE comme un autoencodeur classique ou conserver l'interpolation des codes pendant la dé-

tection d'anomalie. Nos résultats suggèrent qu'il est généralement préférable de ne pas conserver l'interpolation après l'entraînement, mais que l'interpolation — qui est la nouveauté de la méthode — durant l'entraînement améliore significativement les performances du modèle.

**Résultats.** Cette méthode a été testée sur des jeux de données de surveillance vidéo (UCSD Pedestrian, Subway, UCHK Avenue et ShanghaiTech campus) et a donné de bons résultats dans l'ensemble. Sans pour autant dépasser l'état de l'art, c'est la première méthode à être testée sur autant de jeux de données et à réussir aussi bien sur l'ensemble.

De plus, il est à noter que l'IAE peut facilement être combiné avec d'autres méthodes puisque l'on peut le voir comme une simple contrainte sur l'apprentissage d'un autoencodeur.

## Le VIAE

**Motivation.** Cette méthode vise à corriger un des défauts de l'IAE : le fait que l'IAE considère qu'il n'existe qu'un chemin possible entre le présent et le futur.

**Méthode.** Nous intégrons l'IAE dans un cadre probabiliste (aussi appelé cadre variationnel, d'où le nom VIAE). Le VIAE peut modéliser plusieurs chemins et prendre celui qui semble le mieux coller à la réalité.

**Détection d'anomalies.** Pour détecter les anomalies, nous pouvons utiliser le VIAE comme un IAE. Cependant, il est également possible de profiter du cadre probabiliste au delà de l'entraînement. En effet, on peut utiliser le cadre probabiliste pour générer non pas un uniquement présent — à partir du passé et du futur — mais en échantillonner plusieurs. Ensuite, au lieu de calculer une seule erreur de reconstruction — qui nous servira à déterminer si l'observation est anormale — nous prenons la plus basse erreur de reconstruction parmi tous les échantillons générés, en supposant que l'échantillon qui a la plus basse erreur de reconstruction est celui qui a le plus de chance de correspondre à l'observation.

**Résultats.** À l'heure où nous écrivons ce manuscrit, le VIAE n'a été testé que pour détecter des anomalies dans des données de flux réseaux pré-traitées, venant du jeu de

données Kitsune. Dans l'ensemble, le VIAE a donné de bons résultats, parfois significativement supérieurs à l'état de l'art et parfois en dessous tout en restant proche.

Nous avons également le VIAE avec l'IAE sur ces mêmes données. Les résultats de cette expérience suggèrent que le VIAE est belle et bien une amélioration de l'IAE dans la majorité des cas. Des travaux supplémentaires devraient être conduits sur les mêmes jeux de données de surveillance vidéo que l'IAE afin de vérifier si cette amélioration se confirme expérimentalement.

## Le CnC

**Motivation.** Le CnC — pour "Clair et Concis" — est une méthode encore en phase expérimentale qui prend une approche centrée sur la théorie de l'information. La théorie de l'information nous apprend — entre autre — que pour le résultat $x$ d'une variable aléatoire $X$, la longueur $l(x)$ du code optimal $c(x)$ de ce résultat dépend de la probabilité $p(x)$: $p(x) \propto l(x)$. Étant donné que les anomalies sont des évènements hautement improbables dans un contexte donné, nous pouvons faire un parallèle où le contexte est la variable aléatoire $X$ et l'observation est le résultat de cette variable aléatoire $x$.

**Méthode.** Il est important de noter que les détails de cette approche sont encore à améliorer et une grande quantité de méthodes peut être dérivée de ce principe. Voici les principes de base actuellement utilisés :

1. Un autoencodeur est entraîné à compresser-décompresser des données.
2. Un second sous-modèle apprend à masquer le code produit par l'encodeur de manière à ce que la longueur de ce code soit aussi petite que possible sans pour autant trop diminuer la qualité de la reconstruction. L'encodeur de son côté est aussi encouragé à arranger ses codes de manière à faciliter le travail de ce second sous-modèle.

**Résultats et détection d'anomalies.** Pour le moment, cette méthode n'a été évaluée que sur le jeu de données UCSD Pedestrian. Les premiers résultats sont très prometteurs. Cependant, l'entraînement d'un CnC est encore très instable à l'heure actuelle. En effet, l'équilibre — entre la longueur du code obtenu et la qualité de reconstruction — est difficile à atteindre lors de l'entraînement. Lors de nos expériences, nous avons concentrons une partie de nos efforts à résoudre ce problème sans succès systématique.

Concernant la procédure utilisée pour détecter des anomalies, nous avons testé plusieurs solutions: 1) Utiliser la longueur du code (obtenue du second sous-modèle) 2) Utiliser l'erreur de reconstruction et 3) Combiner ces deux valeurs. Les résultats de ce test sont très encourageants. En effet, lorsque l'entraînement s'est bien passé, les 3 solutions sont bonnes. La troisième solution semble supérieure aux deux autres qui elles semblent équivalentes.

Cependant, le résultat le plus intéressant de ce test est le fait que la longueur du code peut-être utilisée seule. Cela suggère que le modèle a effectivement associé des longueurs de code plus élevées aux résultats moins probables, ce qui semble confirmer l'hypothèse que l'on peut modéliser la vraisemblance d'une observation à travers la longueur de son code en modélisant un encodeur optimal.

# Méthodes et résultats secondaires

Nous abordons maintenant les méthodes dont le but n'est pas directement de détecter les anomalies mais d'ouvrir la voie à de nouvelles méthodes de détection d'anomalies.

En particulier, les modèles et sous-modèles issus de ces méthodes pourront ensuite être transférés et utilisés dans des modèles de détection d'anomalies.

## Modèles audiovisuels

Nous avons développé trois méthodes dans l'objectif de modéliser plusieurs modalités de manière jointe, c'est à dire ici de manière à ce que l'entraînement sur la vidéo dépende de l'entraînement sur l'audio et vice-versa. Le contraire serait d'utiliser des dérivés précalculés comme par exemple le flux optique — pour la vidéo — et le spectrogramme — pour l'audio — qui sont calculés indépendamment l'un de l'autre.

**Adaptation de domaine pour la multimodalité.** Le principe de cette méthode, basée sur les travaux de Larsen et al. 2016 et de M.-Y. Liu, Breuel, and Kautz 2017, consiste à considérer les modalités comme des domaines différents qui partagent des informations. Les deux modalités — audio et vidéo — sont projetées dans un même espace de représentation. Cela permet de décoder le code de n'importe quelle modalité avec n'importe quel décodeur, permettant de transformer de la vidéo en audio et inversement. Dit autrement, le modèle "imagine" l'audio qui devrait aller la vidéo (et inversement).

Cependant, on ne peut pas comparer directement deux modalités entre elles pour juger de leur similarité (et donc entraînement un modèle à maximiser cette similarité). C'est là qu'intervient l'ajout de sous-modèles appelés communément "discriminateurs" que l'on trouve habituellement dans les modèles adversariels. Ce sont les experts dont nous parlions plus haut dans le duel incessant entre expert et fraudeur. Ici, les experts servent à estimer si une modalité produite à partir d'une autre semble crédible, ce qui permet aux décodeurs qui produisent ces modalités de s'entraîner à les rendre crédible.

**Modèles à base d'énergie pour la cohérence multimodale.** Dans cette méthode, un autoencodeur sert de discriminateur — d'expert — où son erreur de reconstruction fait office de signal quant à la crédibilité des données. Puisqu'il s'agit d'un système adversariel, le discriminateur est mis face à un générateur — un fraudeur — qui produit de la vidéo et de l'audio.

Ici, l'autoencodeur/discriminateur dispose de 3 encodeurs et 3 décodeurs. Tout d'abord, une paire d'encodeur-décodeurs pour chaque modalité. La dernière paire d'encodeur-décodeurs est utiliser pour compresser-décompresser les codes issus des deux autres encodeurs et mis bout à bout. Cet autoencodeur central sert à la modélisation jointe de l'audio et de la video.

Le discriminateur et le générateur s'entraînent donc l'un contre l'autre, le premier à séparer le vrai du faux et le second à berner le premier. Une fois le modèle entraîné, on peut décider quelques parties du modèle on souhaite utiliser dans un nouveau modèle pour la détection d'anomalies.

**Génération multimodale à base de transformers.** Les transformers sont des modèles dits autorégressifs, c'est à dire qu'à partir d'une séquence de données, ils sont capables de modéliser la suite de la séquence. C'est typiquement ce que fait la prédiction météorologique : elle observe la météo du passé pour déterminer au mieux la météo à venir.

Les transformers ont été initialement développés pour la traduction. Les transformers sont d'abord conditionnés sur une phrase à traduire et produisent le premier mot de la phrase traduite. Puis ils recommencent en regardant de nouveau la phrase à traduire mais cette fois-ci en utilisant le premier mot traduit, et ainsi de suite jusqu'à ce que la traduction soit terminée.

Ici, notre objectif n'est pas de traduire entre deux langues mais entre deux modalités.

Nous nous attendons à ce qu'un tel entraînement soit bénéfique à la représentation jointe de la vidéo et de l'audio. Une fois l'entraînement terminé, comme avec les deux modèles précédents, les différentes parties du modèle peuvent être ensuite utilisées dans des modèles de détection d'anomalies.

## Modèles de méta-apprentissage

Comme leur noms le suggère, les modèles de méta-apprentissage apprennent à apprendre. Cette faculté est particulièrement intéressante pour une éventuelle mise en oeuvre d'une méthode de détection d'anomalies dans le monde réel.

En effet, en fois entraînés à apprendre, ce genre de modèle est capable d'apprendre — ou plutôt de se spécialiser — à partir de très peu d'échantillons. Ainsi, un premier entraînement très général mais coûteux pourra être conduit avant de déployer le modèle dans le monde réel dans de nombreuses situations. À ce moment là, le modèle apprendra rapidement et donc à moindre coût. En bref, le modèle ne réalise qu'un apprentissage coûteux pour tous les contextes au lieu d'en faire un à chaque nouveau contexte comme ce qui est fait traditionnellement.

Nous nous sommes particulièrement intéressés à une catégorie de modèles de méta-apprentissage appelés "réseaux de neurones à mémoire augmentée". Ces modèles sont constitués de deux parties: le réseaux de neurones, comme d'habitude et d'une mémoire externe. Cette mémoire, à l'image d'un disque dur, peut-être effacée pour réinitialiser la spécialisation du modèle. Sur un nouveau contexte, le modèle remplira rapidement cette mémoire. Dit autrement, le réseau de neurones est la partie qui réalise l'entraînement long et coûteux, tandis que c'est la mémoire qui réalise les entraînements rapides et sert à spécialiser le modèle.

Des exemples de tels modèles inclus le MANN (Santoro et al. 2016) et la machine de Kanerva (Y. Wu, Wayne, Graves, et al. 2018). Alors que nous avons obtenu des premiers résultats encourageant avec le MANN, nous n'avons pas été en mesure de répliquer le travail de la machine de Kanerva et n'avons pas de résultats associés.

## Modèles et leurs a priori

Pour terminer les résultats secondaires, nous nous sommes penchés sur l'étude des a priori présents dans la structure des modèles que nous utilisions.

En effet, nous avons débuté cette thèse dans l'optique de réaliser la méthode la plus

générale possible afin que celle-ci puisse être utilisée indépendamment des modalités disponibles. Nous supposions donc qu'il nous fallait une méthode sans a priori également.

Cependant, les travaux de Ulyanov, Vedaldi, and Lempitsky 2018 ont montré que les réseaux de neurones convolutionnels (CNNs) — les réseaux de neurones que nous avons majoritairement utilisés — ont un très fort a priori inscrit dans leur structure.

Mais ces a priori n'ont pas toujours à être considérés comme négatifs, bien au contraire. Ces a priori permettent aux réseaux de neurones comme les CNNs de converger bien plus rapidement vers une solution satisfaisante.

De plus, après plus de réflexion, il semble maintenant évident que l'IAE — notre méthode la plus fructueuse dans cette thèse — a un très fort a priori sur les séquences temporelles et que c'est justement cet a priori qui lui permet de se hisser au niveau de l'état de l'art.

Cela ouvre donc des possibilités pour chercher à produire des a priori structurels comme ceux des CNNs ou des IAEs afin d'améliorer la détection d'anomalies.

Nous notons également que nous avons été en mesure de répliquer aisément les travaux de Ulyanov, Vedaldi, and Lempitsky 2018. Les résultats pratiques de cette publication, bien que surprenant et impressionnants — comme une bonne reconstruction d'images corrompues, la super-résolution et le défloutage —, sont simples à reproduire.

## Apprentissage de représentations

Nous arrivons au sujet de l'apprentissage de représentations. Nous pensons qu'il est indispensable d'explorer ce domaine afin de créer des modèles satisfaisants de détection de comportements humains anormaux lorsque que ces comportements deviennent complexes (*e.g.* contrairement à la plupart des comportements observés de loin par une caméra de surveillance).

### Mesures de distance dans l'espace d'entrée

Tout d'abord, il est important que la majorité des travaux de détection d'anomalies utilisant de l'apprentissage profond utilisent des autoencodeurs et donc l'erreur de reconstruction pour déterminer si une observation est anormale.

Cette erreur de reconstruction est presque toujours calculées dans l'espace de la modalité. Pour une image, cela revient à calculer la différence d'intensité des pixels par exemple. À partir de là, il est très facile d'imaginer des scénarios où une image visuellement iden-

tique - à l'exception de quelques pixels — soit considérée comme plus différente de sa reconstruction — et donc plus anormale — qu'une autre image qui n'est pas visuellement similaire à sa reconstruction mais qui produit tout de même une erreur de reconstruction plus faible.

Cet aspect rejoint le fléau de la dimensionalité, qui nous explique que les mesures de distances dans les espaces à haute dimensionalité — comme les images, le son ou encore la vidéo — n'a pas de pouvoir discriminant.

L'un des intérêts d'apprendre des représentations est de projeter ces observations depuis leurs espaces éparses à haute dimensionnalité dans des espaces denses où les mesures de distances retrouvent du sens.

### La notion de comportement est abstraite

Nous pensons que cela est particulièrement vrai pour les comportements humains, puisque cette notion est elle-même abstraite.

En effet, lorsque l'on considère les comportements humains, il devient très difficile de tracer une ligne ou de donner des critères qui séparent de manière non ambiguë deux comportements similaires en apparence.

Cependant, apprendre des représentations de comportements humains nous donne également l'occasion de mieux les séparer dans l'espace de représentation en développant des méthodes qui ont cet objectif.

Il est également envisageable d'utiliser des modèles de représentation afin d'améliorer l'interprétabilité des décisions du modèle, un aspect qui fait généralement défaut aux modèles à base de réseaux de neurones. En effet, avoir cet espace de représentation nous donne également l'opportunité de développer des modèles qui font le travail inverse, c'est à dire générer des échantillons à partir de représentations qui pourraient expliquer pourquoi le modèle a décidé que tel ou tel échantillon est anormal.

### Modèles de représentations

Dans cet objectif, nous recensons quatre modèles d'apprentissage de représentations.

**Autoencodage au-delà des mesures de similarité sur les pixels** Le premier modèle, développé par Larsen et al. 2016 et communément appelé "VAE-GAN", utilise les représentations apprises par le discriminateur — l'expert — afin de calculer une nouvelle mesure de similarité.

En pratique, les données générées par le fraudeur sont projetées dans un espace de représentations appris par le discriminateur et les données sont comparées dans cet espace là. Les échantillons autoencodés par cette méthode semblent plus visuellement similaires — d'un point de vue subjectif et humain — qu'avec les méthodes précédentes, ce qui semble correspondre à ce que nous cherchons.

**BYOL** La méthode BYOL (Grill et al. 2020) quant à elle prend une observation et la transforme de deux manière différentes, créant deux "vues". Par exemple, ces deux vues peuvent correspondre à deux rognages différents d'une même image. Ces deux vues sont — pour faire simple — données au même encodeur et il est demandé à cet encodeur de produire la même représentation malgré le fait que les deux vues soient différentes parce que ces deux vues représentent la même observation.

D'autres contraintes — pour lesquelles nous n'entrerons pas plus en détail ici — sont ajoutées pour empêcher le modèle de tout simplement projeter toutes les vues sur un seul et unique point dans l'espace de représentation.

Les encodeurs produits par la méthode BYOL peuvent ensuite être utilisés dans d'autres tâches. En particulier, les auteurs de BYOL ont évalué leur modèles en appliquant un simple algorithme d'évaluation linaire sur les représentations qu'ils ont obtenues afin de classifier les images du jeu de données ImageNet.

**DINO** La méthode DINO (Caron et al. 2021) est en apparence très semblable à la méthode BYOL. Cependant, sans trop entrer dans les détails, la méthode DINO est capable d'obtenir de meilleurs résultats que la méthode BYOL sur la même tâche sans pour autant avoir besoin de générer deux vues de la même observation.

Cette différence est très importante dans notre cas, puisqu'il aurait fallu spécifier les transformations qui produisent ces vues pour chaque modalité. Or, dans cette thèse nous visons la méthode la plus générale possible quant à l'adaptation à de nouvelles modalités.

**Caractéristiques audiovisuelles multi-sensorielles** Dans les méthodes d'apprentissage de représentations que nous présentons ici, celle développée par Owens and Efros 2018 est la seule multimodale.

Le principe est le suivant:

1. Premièrement, encoder séparément la vidéo et l'audio

2. Deuxièmement, mettre bout-à-bout les codes ainsi obtenus et les encoder de nouveau
3. Enfin, appliquer un dernier encodeur qui ne produit qu'une seule valeur, que nous appellerons "énergie".

Une fois sur deux pendant l'entraînement, lorsqu'une observation audiovisuelle est montrée au modèle, la piste audio est décalée — vers l'avant ou vers l'arrière — et le modèle doit déterminer si la piste audio a été décalée. Le modèle est entraîné à produire des valeurs d'énergie positives si la piste audio a été décalée et des valeurs négatives si la piste audio est alignée avec la piste vidéo.

Une fois l'entraînement terminé, le quatrième encodeur est retiré, afin de ne garder que les représentations jointes. Owens and Efros 2018 ont ensuite évalué leur méthode sur des tâches en aval comme la séparation des éléments sonores qui sont présents sur la vidéo de ceux qui ne le sont pas (*e.g.* retirer une voix off). Les auteurs ont obtenu d'excellents résultats sur toutes les tâches en utilisant les représentations jointes apprises.

# Discussion

Maintenant que nous avons présenté le domaine dans lequel cette thèse est ancrée ainsi que les méthodes et résultats de cette thèse, nous résumons nos découvertes et en tirons des conclusions.

**Définir les anomalies dans un contexte.** Nous avons vu qu'il faut prendre garde à définir ce qui est normal, et non ce qui est anormal — dans un contexte donné — si l'on souhaite détecter correctement des anomalies.

Nous avons également vu qu'il est crucial de faire attention à ne pas marginaliser des comportements normaux mais sous-représentés dans les données à notre disposition comme cela a déjà le cas en pratique.

**La multimodalité au service du contexte.** Nous avons également souligné que l'intérêt de la multimodalité est actuellement sous-évalué. En particulier, des informations supplémentaire sur le contexte à un moment peuvent être nécessaires (*e.g.* l'heure, la date, le contexte sanitaire, *etc.*).

**Avoir de bons a priori.** Nous nous sommes rendus compte que les a priori dans nos méthodes sont plus présents qu'il n'y parait au premier coup d'oeil. Nous avons également observé que certains a priori structurels peuvent être très bénéfiques à la détection d'anomalies.

**Modéliser les comportements humains comme des mécanismes temporels.** Dans cette thèse, nous avons notamment utilisé des a priori sur les séquences temporelles afin de mieux modéliser les comportements humains.

Bien qu'il reste une grande quantité de travail sur la détection de comportements humains anormaux, ce choix de modéliser les comportements comme des mécanismes temporels a donné de bons résultats et nous apparaît comme une piste solide à poursuivre.

**La modélisation multimodale encore à un stade précoce.** En nous penchant sur la littérature d'apprentissage machine — en particulier sur la détection d'anomalies et sur le comportement humain — nous nous sommes rendus compte le domaine de la modélisation jointe de plusieurs modalités est encore très jeune.

D'après nos travaux, nous concluons qu'il est nécessaire de d'abord avancer sur ce domaine afin de pouvoir ensuite mieux avancer sur la détection multimodale de comportements humains anormaux.

Nous avons vu que l'apprentissage de représentations est un domaine en pleine croissance qui peut justement nous aider à avancer sur la modélisation multimodale. En particulier, nous attendons de l'apprentissage de représentations des modèles qui produisent des représentations multimodales spécifiques aux comportements humains.

Par ailleurs, nous avons vu que, de part la nature auto-supervisée de l'apprentissage de représentations, la quantité de données que nous pouvons exploiter pour cette tâche est bien plus grande qu'il n'y semble au premier abord.

**Le monde réel évolue en permanence.** Pour finir, nous avons vu que la nature changeante du monde réel pose un challenge important. En effet, en général les modèles d'apprentissage profond pour la détection d'anomalies sont entraînés une fois avant d'être mis en production. Les mettre à jour ensuite demande un ré-entraînement complet.

Nous avons identifiés quelques pistes prometteuses afin de pallier à ce problème, comme les modèles de méta-apprentissage.

# Travaux futurs

Nous nous tournons maintenant vers l'avenir et les perspectives que nous avons identifiées par rapport à la détection multimodale de comportements humains anormaux.

**Perspectives sur l'apprentissage de représentations.** Nous anticipons de futurs travaux sur l'apprentissage de représentations sur les trois tâches suivantes :

1. Améliorer les représentations internes aux modèles des concepts du monde réel.
2. Améliorer l'interprétabilité des futurs modèles, en créant une relation entre les représentations et les concepts du monde réel auxquels elles correspondent.
3. Diminuer l'écart entre les domaines de l'apprentissage de représentation, du comportement humain et de la détection d'anomalies.

En particulier, nous anticipons le développement de modèles de représentations de comportements humains qui exploitent les a priori sur le comportement humain et les séquences temporelles.

**Perspectives sur les jeux de données.** Pendant nos travaux de thèse, les jeux de données ont toujours posé de nombreux problèmes vis-à-vis de nos objectifs.

Nous estimons que des travaux devront être conduits afin de constituer de nouveaux jeux de données pour répondre à ces problèmes. En plus de jeux de données variés et complets pour évaluer les méthodes de détection de comportements humains anormaux, nous attendons les améliorations suivantes à l'avenir:

1. Des jeux de données pour évaluer la qualité des représentations apprises.
2. Les futurs jeux de données devraient s'assurer d'être réellement représentatifs du monde réel et non d'une vue restreinte du monde réel.
3. Les futurs jeux de données pour la détection de comportements humains anormaux devraient présenter des gradients de difficulté. Les difficultés plus basses servant à réaliser des preuves de concept et les difficultés plus hautes servant à valider les méthodes, leur robustesse, *etc.*
4. Des jeux de données dits "privacy-preserving", c'est à dire qui communiquent aussi peu d'informations personnelles que possible, permettraient un accès à une quantité significativement plus importante aux données, établissant des jeux de données de meilleure utilité.

**Perspectives sur les méthodes adaptatives.** Enfin, nous prévoyons que des travaux sur les méthodes qui s'adaptent aux changements du monde réel verront le jour, et ce dans l'objectif d'une mise en production plus fiable sur le long terme.

En particulier, les travaux existants sur les modèles de méta-apprentissage — dont les modèles à mémoire augmentée — semblent prometteurs et appellent à de nouveaux travaux dans le domaine de la détection d'anomalie. Notamment, ces travaux devront être capables de tenir compte des retours utilisateurs afin de pouvoir corriger leurs erreurs.

# Table of Contents

Chapter 1

# Introduction

We first lay foundations required to understand the remainder of this document. In this section, we will start by explaining the motivations that led to this thesis and the problems we aim to solve. Then, we will give a thorough definition of the task we are tackling, namely Anomaly Detection. After that, we will restrict to anomalous human behavior detection, the sub-field of Anomaly Detection we are interested in. We will then talk about the multimodal component of this thesis and go over the different modalities we can exploit for our problem. To conclude this introduction, we will briefly go over the preceding thesis, which is the work of Fayet Cédric, and how it relates to the present thesis.

## 1.1 Motivation

In this subsection, we start by explaining the different aspects of the general problem motivating the present work, and how these aspects interleave. Then, we will present some real-world examples of this problem in order to underline the broadness of the problem and its relevance.

### 1.1.1 General Problem Overview

Human beings continuously exhibit behaviors reflecting their internal state and their internal cognition. Depending on these internal states, internal cognition and their potential consequences, one might estimate it is important, or even vital sometimes, to react to them. Thus, human behaviors carry relevant information for their observers.

In particular, anomalous human behavior are a special set of behaviors which, when they are witnessed, most likely call for action. Due to the scale of human population and the usefulness of detecting anomalous human behavior, this makes it very desirable to automate the detection of anomalous human behavior and also detect anomalous human

behavior less obvious to the human observer.

Human behaviors can be observed through a variety of modalities. The two that usually come to mind are vision and hearing, but human behaviors can also be observed through other modalities. Here are a few examples: text on social networks, body signals (e.g. heart rate) or even smell to an extent.

Those modalities usually suffer from a problem called *the curse of dimensionality*, a problem that arises when data is in high-dimensional spaces. For example, an image with a fairly low resolution (128x128) already has a dimensionality of 16 384, which is far more than the 3-dimensional space humans are used to. This makes the development of machine learning models required for automating anomalous human behavior detection.

### 1.1.2 Application cases

Cognition occurs at all time, even during sleep, and produces human behaviors. For this reason, humans always exhibit some behavior that is the conscious or unconscious result of cognition. This is why there are probably as many potential real-world applications of anomalous human behavior detection than there are situations involving humans. Moreover, these real-world applications are as diverse as their corresponding situations, here are only a few examples:

#### 1.1.2.1 Security

Security is usually the first application that comes to mind. Whether we are in a private or a public place, it is desirable to detect anomalous human behavior in order to prevent incidents such as assaults or dangerous angry outbursts, for both voluntary and involuntary incidents. A more specific, up-to-date, example would be the prevention of terrorist assaults in airports.

#### 1.1.2.2 Piloting

For some vehicles, their operation can be delicate and require the full focus of their pilot. Flying a fighter aircraft requires the pilot to be in full possession of their reflexes, while a pilot flying a commercial airliner may be responsible of many lives in the plane. Thus, it can be useful to ensure that those pilots are in full control before boarding or during the operation. anomalous human behavior detection would allow for automatic detection of impairing factors, such as stress or sleep deprivation. Another example would

be the driving of a construction machine, which has the potential for a lot of damages, both material and human, if driven improperly. This principle can also be extended to different kinds of devices, such as electric saws or electrical cabinets.

#### 1.1.2.3 Mental Health

Some applications of anomalous human behavior detection also take into account long-term behaviors, and this is the case for applications in the mental health field. Nowadays, people put large amounts of personal data on social networks. Such amounts of personal data give insights about a person's mental health and could be useful in preventing or diagnosing mental health issues, like depression. The same idea could be used for detecting a person's suicidal tendencies and helping them.

#### 1.1.2.4 Invisible disabilities

Somewhat related to mental health, hidden disabilities are disabilities hard to detect, this is mostly the case for invisible disabilities (or hidden disabilities). Invisible disabilities primarily include neurological disabilities but can also be of physical nature, examples for such disabilities include some heart conditions, fibromyalgia, diabetes, sleep disorders, neurodevelopmental disorders, etc. Invisible disabilities incur behavioral differences, making them a suitable target for anomalous human behavior detection. A more specific example would be the detection of Autistic Spectrum Disorders (ASD) in the early life of a child in order to provide help, first during the child's development and throughout life in general.

#### 1.1.2.5 Network

A part of the global network traffic is a direct result of human activity. Thus, in some cases, human behaviors can be observed through network activity. For a network security administrator, anomalous human behavior detection can be useful if normal behaviors correspond to intended usages of the network. An anomalous human behavior detector would then raise alarms when suspicious or unintended usage is being made of the network. In a similar way, undesirable non-human activity (e.g. DDoS attacks or automated intrusions) could be detected by such a system, as non-human behaviors would not be considered as normal human behaviors.

#### 1.1.2.6 Electronic voting

We take a broad definition of electronic voting, where a user can either vote for or rate an entity (e.g. an item, a movie or a candidate). Systems like this will usually put conditions before voting, such as a proof of personhood, a proof of work or a proof of purchase. These conditions try to ensure that a single person will not have more that one vote, but these conditions usually have two flaws. First, these proofs cannot always guarantee the uniqueness of a user's vote. Second, these proofs restrict access to voting to users that sometimes would be legitimate because those users cannot provide said proofs.

A common example of this is online shopping, where users can rate a product after buying it, and where both flaws can be observed. First, many cases of frauds from the seller or its competitors have been documented (e.g. a seller paying costumers to buy the product and give a positive rating). Second, a costumer having purchased an item will likely not be able to give ratings on a different website for the exact same item.

Tournesol Hoang 2021 is an example of a collaborative system where users give different ratings to Youtube videos in order to promote ethical and important videos. They currently use a proof of personhood via the requirement of having an institutional email address, which greatly restrict access. Moreover, it is possible to have several institutional email addresses. Ideally, a more accessible authentication system would be preferred but this would also open the door to malicious attacks, for example allowing malicious communities to promote unethical videos. In this regard, an anomalous human behavior detector would help to prevent and remove ratings from such attacks.

Of course, anomalous human behavior detectors could also be considered for democratic applications such as presidential elections, to help improving security of electronic voting which is critical in such cases.

#### 1.1.2.7 Summary

These examples illustrate an important point: Applications for anomalous human behavior detection are numerous and diverse. anomalous human behavior detection is interesting for a wide range of sectors that may seem unrelated, including airports, medical offices, construction sites, social networks, grocery stores, etc.


Figure 1.1 – Examples of real-world application fields for anomalous human behavior detection.

## 1.2 Anomaly Detection

In this subsection, we will first see that defining what is an anomaly is not straightforward at all. Then, we will see that this difficulty to define what should be considered anomalous often creates a reality-expectation mismatch, between what is asked of anomaly detection, and what it actually achieves. Finally, we will see a side of anomaly detection which is often neglected but can improve detection, including by reducing the impact of the mismatch between expectations and reality.

### 1.2.1 What is an anomaly ?

First, let us turn toward how anomalies are defined in the literature. In their survey on anomaly detection, Chandola, Banerjee, and Kumar 2009 define anomalies as

> [...] patterns in data that do not conform to a well defined notion of normal behavior.

In his book "Identification of outliers", D. M. Hawkins 1980 gives a definition for

outliers, a term similar to anomalies:

> An outlier is defined as an observation, which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.

In both cases, there is a notion of mechanisms (or behaviors) responsible for generating data. Some of these mechanisms are normal, and some are anomalous. Thus, anomalies are samples generated by anomalous mechanisms.

However, the most important thing to notice is that anomalous mechanisms are not directly defined. Rather, they are indirectly defined by opposition with normal mechanisms.

This distinction is essential to grasp a central concept in anomaly detection. We cannot directly define anomalies, we do not know them and we cannot enumerate them: Anomalies are unexpected and cannot be predicted. The same goes for their generating mechanisms.

Instead, we must define all that is normal: normal patterns, normal mechanisms, normal samples... Ideally, we would be able to enumerate absolutely all normal mechanisms.

#### 1.2.1.1 Context is needed

The notion of normal mechanism means nothing if we do not introduce the notion of context. Indeed, a sample (or a mechanism) can only be normal relatively to a given context. This context can be simply defined as a set of normal mechanisms. The notion of context is helpful to see that a slight change in the set of normal mechanisms produces a new context. We provide two examples to help grasp the importance of the notion of context, a rather abstract one and a concrete one.

The first example, presented in Figure 1.2, shows a simple set of small numbers. Looking at the normal samples, we can estimate that there is only one normal mechanism which generates only even numbers. In this case, 7 could not have been generated by a normal mechanism and thus is anomalous. This example also relates to the game called "Find the odd one out", where the player has to find the mechanism that generated all samples except one, in order to isolate the anomaly.

In the second example, presented in Figure 1.3, we show a simplified task of anomalous human behavior detection in order to underline the importance of context. In the first situation, which is a party, normal behaviors are of a rather happy nature. In this situation, someone expressing their joy is completely normal and somewhat expected. However, the same behavior can be anomalous and considered unsettling in the context of a funeral.

4 2 8 16 7 10 6

Figure 1.2 – A simple example of anomaly detection where the context is easy to model.

You would not expect someone to be so cheerful they sing and dance during a funeral eulogy. Saying you would be surprised is probably an understatement.

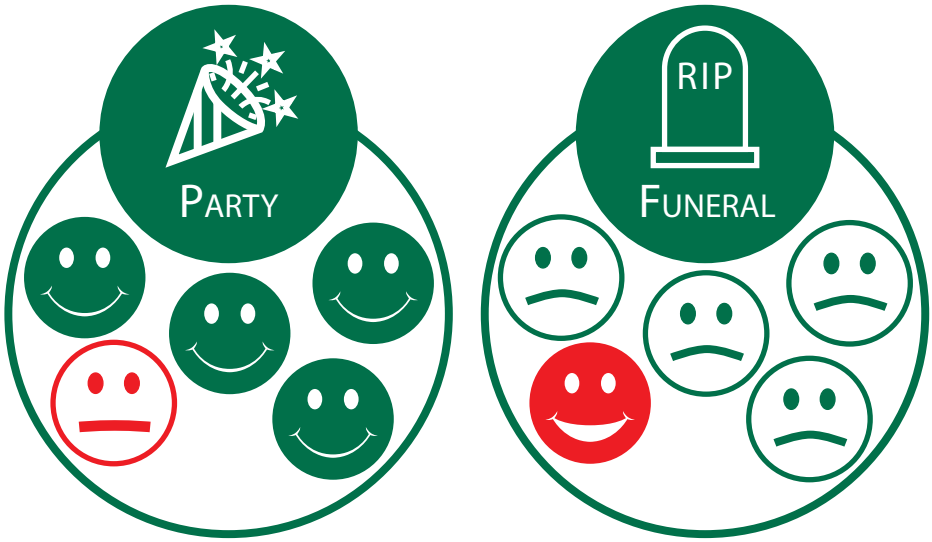


Figure 1.3 – Two contexts are presented. For most normal behaviors in the first context, they are considered anomalous in the second one, and vice versa. Thus, a sample is always anomalous relatively to a context.

#### 1.2.1.2 It never is simple

Now that we have seen that we need context to decide if a sample is anomalous, let us look back at the numerical example presented in Figure 1.2. Do you think the number 17850 is anomalous in this context ? And why ?

You cannot be sure the normal generating mechanism is "1) Any even number", or "2) Any even number with 1 or 2 digits". You cannot even be sure the normal generating

mechanism is not "3) Any number in $\{2, 4, 6, 8, 10, 16\}$". However, what is the more likely rule, n°1, 2 or 3 ?

Rule n°3 is oddly specific, and would make for a poor "Find the odd one out" game as any sample could be considered anomalous by constructing rules as subsets containing all samples except one. The prior probability of this rule is extremely low.

Both rules n°1 and 2 seem reasonable and we can safely assign them similar prior probabilities. However, assuming a uniform distribution when drawing from either rule n°1 or 2, the result $\{2, 4, 6, 8, 10, 16\}$ is significantly more likely for rule n°2 than rule n°1.

But, we can never be sure. For all we know, rule n°1 (any even number) could very well be the underlying rule generating our context. For now, let us assume that rule n°2 is the only normal mechanism of our context, and ask a difficult question: was 2 generated by this rule ?

2 is not very likely for rule n°2 (2%), but it could have been generated by rule n°1 or 3 and these rules (n°1 and n°3) would be anomalous mechanisms in this case. As we have established, rules n°1 and 3 are less likely than rule n°2 but the probably they generated observation 2 is not equal to 0. In addition, while the number of normal mechanisms is limited, the number of anomalous mechanisms is not. Many other anomalous rules could explain this 2, but all those rules are less likely than our normal rule, making rule n°2 the better explanation still.

In general, any normal sample has a chance to have been generated by any anomalous mechanisms. We provide visual examples using two Gaussian distributions in Figure 1.4 and 1.5. In Figure 1.4, the distributions overlap, illustrating situations where there exist values which could have been generated by either mechanism. Figure 1.5 is a special case, derived from the one presented in Figure 1.4, where the two distributions can be disentangled if we have access to an additional feature. While this special case is extremely unlikely in real-world applications, additional features still have the potential to help discriminating anomalies.

This means we will likely never build an anomaly detection system with perfect separation between normal and anomalous samples in a real-world setup, where anomalous mechanisms cannot be enumerated. Instead, uncertainty should be embraced rather than fought, in order to provide better informed decisions. Indications of high uncertainty should then be treated as indications to investigate further, to fetch features viewed as useful for the current problem, in order to reduce uncertainty.

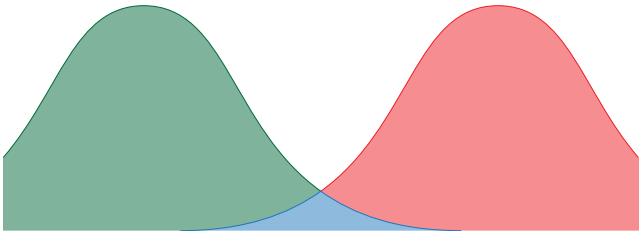

Figure 1.4 – Example of two mechanisms whose distributions overlap when using a single feature for each sample. The green distribution represents a normal mechanism and the red one represents an anomalous mechanism. Vertical axis denotes likelihood and the horizontal axis denotes the value of the feature.

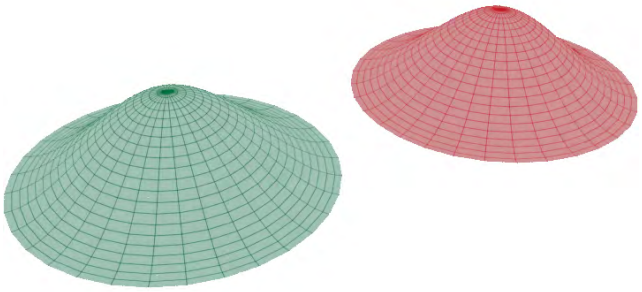

Figure 1.5 – The same example as in Figure 1.4, presented with two features. The addition of a feature allows us to disentangle the two distributions in this case and remove uncertainty. Height denotes likelihood and the axes of the horizontal plane denote the values of the features.

### 1.2.2 Warning about eventual tool/task mismatches

As we have said before, anomalies are indirectly defined by opposition to normal samples. However, this way of thinking is not intuitive and commonly creates mismatches between what anomaly detection does, and what it is asked to do. For example, one might want to use an anomaly detection system to detect violent behaviors. This is not really a task for an anomaly detection system, but a classical classifier, where both classes — 1) non-violent and 2) violent — are clearly identified.

By definition, anomalies are rare and unpredictable. The likelihood to have all of them in the dataset is extremely low. Thus, an anomaly detector may wrongly consider some out-of-distribution samples to be anomalous. Taking our previous example, it is unlikely

that all possible non-violent behaviors would be present in the dataset, in particular rare non-violent behaviors. For an anomaly detector, those non-violent behaviors would be as unexpected as violent behaviors would be, raising then what would be a false alarm for the given task.

However, in this setup, it is still possible to use the anomaly detector as an auxiliary system to a general classifier. In this case, the anomaly detector would serve as a first filter, feeding only anomalous samples to the main system, simplifying its task.

The important thing to remember is that anomaly detectors are one-class classifiers, meaning only one class is known: the normal class. Therefore, they should not be used as multi-class classifiers — 2 usually — and the user should make sure their task is in adequacy with anomaly detection.

### 1.2.3 Active Anomaly Discovery

So, let us say you built an anomaly detection system, trained it on the normal data at your disposal, and you now put it in production, in the real world. After some time, your model raises a false alarm, what do you do ? You likely trained your model for hours, or maybe even days or weeks. And even if you were to retrain your model, you have little guarantee that your model will change its decision: it only received one more sample of a normal mechanism, among possibly tens of thousands of samples from many normal mechanisms. The same goes for false negatives.

This is the problem Active Anomaly Discovery (AAD) (Das et al. 2016) — an application of active learning for anomaly detection — is trying to solve: How to efficiently and robustly incorporate user feedback in order to rectify false positives/negatives for a system already in production.

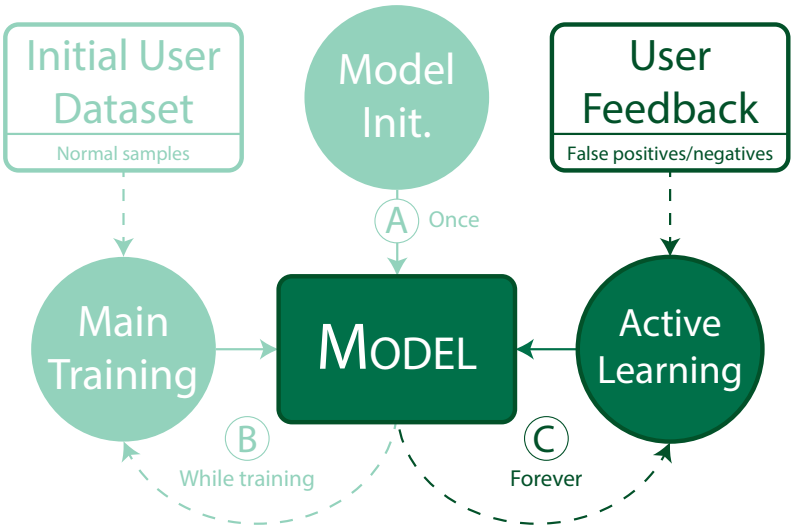


Figure 1.6 – Illustration of the addition of active learning in the global training pipeline. A) The anomaly detection model is initialized and B) trained on the initial user dataset until some criteria is met. The model is then put in production and C) updated according to user feedback during production.

We illustrate the addition of active learning in Figure 1.6. Once the model is put in production, the intended user can provide feedback by flagging false positives and false negatives. True positives and false positives can also be added to the user's dataset, in order to improve the dataset for future complete retraining of the model or even switching to a new model altogether.

It is worth noting that one goal of deploying an automated system is to reduce human labor and this is particularly true in anomaly detection. Thus, feedback should not be asked for each sample the model evaluates but rather a tiny fraction of the observed data. This makes the selection of the samples for which feedback will be requested an important part of active learning, and therefore of AAD.

## 1.3 Anomalous Behaviors Detection

Now that we have talked about anomaly detection in general, we narrow down on the sub-field of Anomalous (Human) Behavior Detection, which is the main focus of this thesis.

As we have seen before, anomalies do not fit a given definition of normality. This is true as well for anomalous behaviors. The key difference here is the introduction of the notion of human behavior. On one side, a human behavior can be defined as a mechanism, resulting from internal cognition, which produces a sequence of observable data about a human (such as actions and external states). On the other side, specific instances of human behavior are not so clear cut (e.g. calmness, tiredness, etc.).

For example, impatience can be mistaken for arrogance, irony for sarcasm, etc... It is quite hard to define clear cut boundaries to delimit behaviors. We make an analogy with the paradox of the heap illustrated in Figure 1.7. This paradox asks when does a heap of sand stops being a heap when its grain are removed one by one and exposes the vagueness of the word heap.

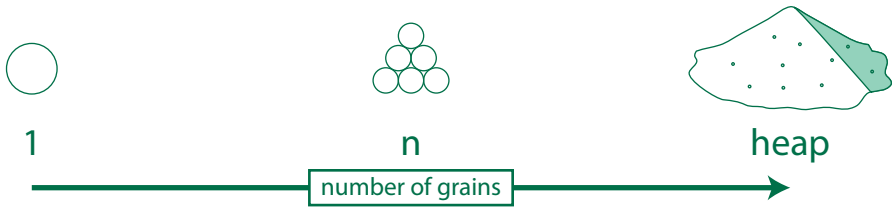


Figure 1.7 – Illustration of the paradox of the heap. One grain of sand is not a heap, and two grains of sand are not either. If we keep adding grains of sand, when does it become a heap ? We can also do the opposite: remove grains of sand from a heap and ask ourselves when does it stop being a heap.

We can ask this question for many, if not most, behaviors. If we see two children fighting, are they actually fighting or are they reenacting a scene from a cartoon, with no intent or risk to hurt each others ? From the point of view of the observer, when does one becomes the other ? The same question could be asked for two adults pretending to fight in public for some reason (e.g. a prank, an artistic show, etc...)

But the most important is that we do not share the exact definition of those words (i.e. violent, pretending, artistic, etc...) as everyone else. This is important because humans are at some point going to label data as either normal or anomalous, for making the training dataset which will represent normal behaviors or during production when providing user feedback.

In the Emo&ly dataset (Fayet et al. 2018), which we will talk about later, humans were asked to "annotate reactions that are unexpected". This is probably the main reason

why the labels differ so much from an expert to an other, as experts did not share a common and precise definition of "unexpected".

Significantly increasing the number of experts for labeling behaviors might help solve this problem. However, this would as well significantly increase human labor needed which is not often possible.

Therefore, users of anomalous human behavior detection systems should be aware of the vagueness induced by the definitions of different behaviors and should aim to provide a definition of normality as clear as possible.

### 1.3.1 Under-represented behaviors and the definition of anomalous behaviors

Machine Learning often aims at modeling the real world as an end goal. Many efforts have been made to increase the size of datasets for example, so that data collected becomes more and more representative of the real world. However, regarding humans, the real world itself is not perfect. Thus, if we model accurately the real world, we will incorporate undesirable biases such as racism. We do not want to automate racism.

There already are several reports of real world cases where such biases were found in large scale applications. Twitter's algorithm for automated image cropping was found to have a significant bias towards centering on white males (Yee, Tantipongpipat, and Mishra 2021). Of course, this was not intended by Twitter but not accounting for this bias in their data led to this situation.

Another example of such problem was found in a widely-used algorithm responsible for managing the health of populations, affecting millions of patients (Obermeyer et al. 2019). In this case, the algorithm used the estimated health care costs of a patient as an heuristic to determine a risk score associated to this patient, the logic behind being that the sicker a patient is, the more costly it will be to treat them. However, as black patients tend to have lower income, they also tend to not be able to spend as much money in healthcare. Thus, at similar sickness level, the algorithm will tend to give higher risk scores to white patient than to black patients. Obermeyer et al. 2019 estimated correcting this bias *would increase the percentage of black patients receiving additional help from 17.7 to 46.5%.*

In a similar way, some behaviors may be under-represented in a dataset, not because they occur significantly less often, but because of the ways the dataset was built. With

this in mind, it becomes obvious that any anomalous behavior detection system deployed in the real world should ideally incorporate some form of active learning to rectify biases that were not accounted for in the training set and better represent missing or under-represented normal behaviors to the model.

## 1.4 Multimodal Anomaly Detection

Human behavior can be observed through a wide variety of modalities, such as video, audio or text. In this sub-section, we will review seven modalities which can help us detect anomalous human behavior and for some of them, we will see a few derivatives of these modalities for specific situations.

It is important to note that these modalities are not mutually exclusive and can be modeled together to improve detection. In some cases, such as audio and video, it is even possible that both modalities were produced by a single mechanism (or behavior). This behavior must then be contained in the mutual information between these modalities.

### 1.4.1 Video

Video is usually the first modality we think of when we are talking about human behavior. This modality covers a wide range of situations, from surveillance cameras to face-cam videos on websites such as YouTube or Twitch, or virtual conferencing software such as Skype or Zoom. This also includes situations such as someone broadcasting the stream of their phone's camera, on which violent scenes could be observed for example.

Video has been extensively studied pretty much since we have been able to connect a camera to a computer. This led to the development of many techniques for extracting information from a video stream. We will review some common ways to extract information from videos in a way that can be useful for detecting anomalous human behavior.

#### 1.4.1.1 Raw video

The default video stream. The data usually takes the form of a tensor with 4 dimensions: time, height, width and channels. Channels are most often expressed in grayscale or RGB (red, green, blue). Depth can be added to the RGB channels, resulting in a RGB-D video. The next techniques are usually computed from the raw video stream.

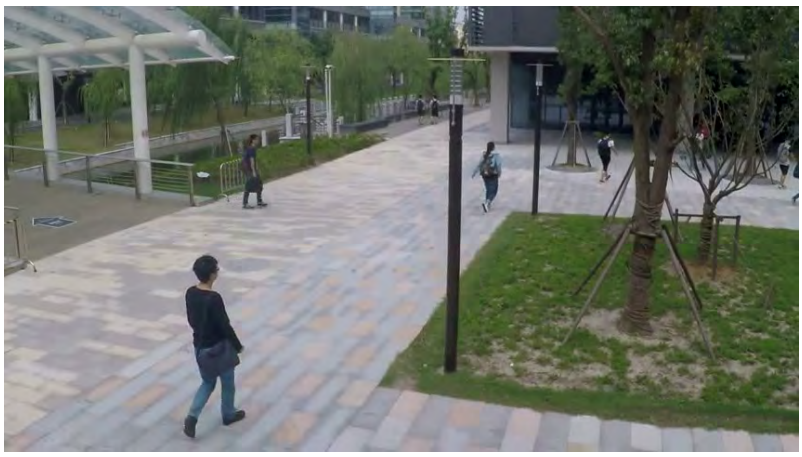

Figure 1.8 – Example of a single surveillance video RGB frame from the ShanghaiTech Campus dataset.

#### 1.4.1.2 Optical Flow (Horn and Schunck 1981)

Computed from two consecutive grayscale frames from a video stream, the optical flow is typically a tensor with at least 3 dimensions: height, width and channels. There are only two channels: amplitude and angle of the movement. If one compute optical flow for every pairs of consecutive frames, we now have a fourth dimension (time). Several improvements were made to Optical Flow, including the deep learning variants FlowNet (Dosovitskiy, Fischer, et al. 2015) and FlowNet 2.0 (Ilg et al. 2017).

Optical Flow focuses on movement, retaining little information about shape and color, making it particularly useful for detecting anomalous trajectories.

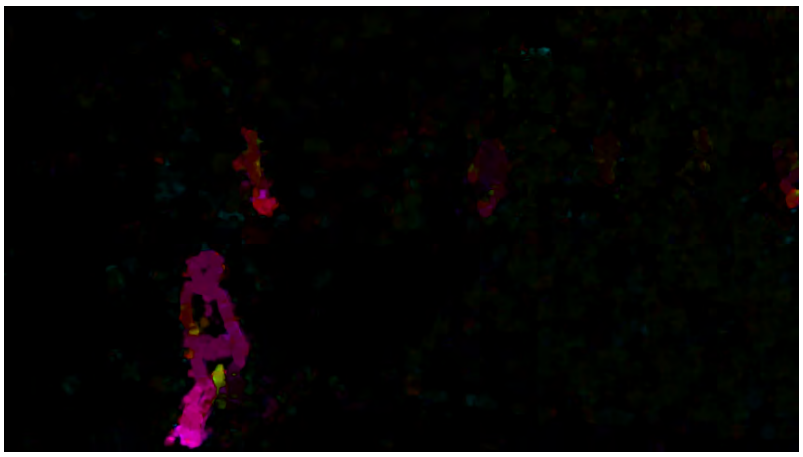

Figure 1.9 – Optical flow computed from two frames at the same moment as in Figure 1.8 was taken. Color hue indicates the angle of the movement (red is directed to the right), while color value indicates the speed of the movement.

#### 1.4.1.3 Landmarks

Whether we are interested in seeing the full body or only the face, we can take advantage of defining landmarks on specific positions. For example, one might want to pinpoint body joints such as shoulders, hands, feet, knees, and so on... While the dimensionality of a video frame is extremely high (height times width times channels), the dimensionality of landmarks is much more manageable (number of landmarks times 2). Landmarks estimation methods usually yield between 20 and 40 landmarks for body joints and 60 to 80 for facial landmarks.

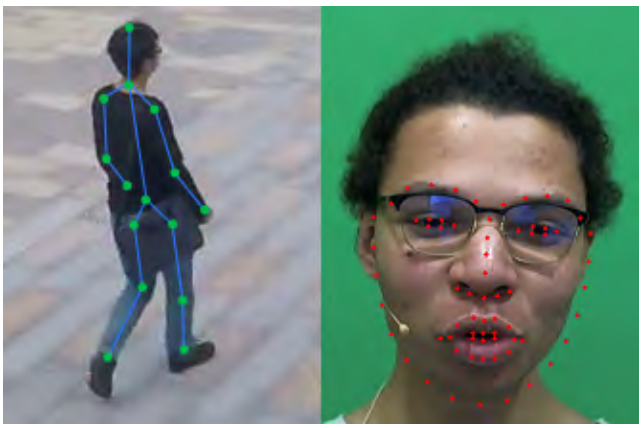

Figure 1.10 – Left) Landmarks for body joints on the previous example in Figure 1.8. Right) An example of facial landmarks detection using DLib.

#### 1.4.1.4 Difference of Gradients/Flows

A common technique in image processing is the usage of Difference of Gaussians, where the difference between an image and a blurred version of the same image is computed. This process is useful for revealing edges. Difference of Gradients takes one more step, as they are the difference between the Difference of Gaussians computed on two consecutive frames independently. Figure 1.11 shows an example of Difference of Gradients.

In a very similar way, Difference of Flow is the difference between two Optical Flow computed from two consecutive pairs of frames. For this, the difference is computed using Cartesian coordinates, and then is translated to polar coordinates as before (yielding the amplitude and angle of the movement).

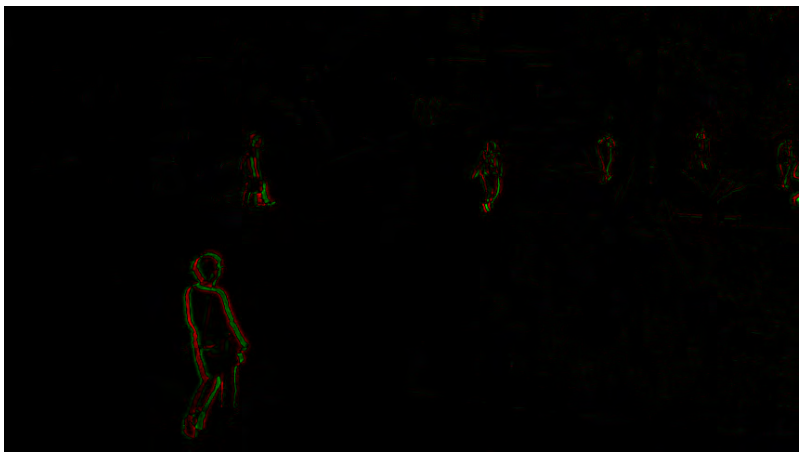

Figure 1.11 – Difference of Gradients computed from two frames at the same moment as in Figure. For better visualisation, negative values were put in red and positive values in green. 1.8

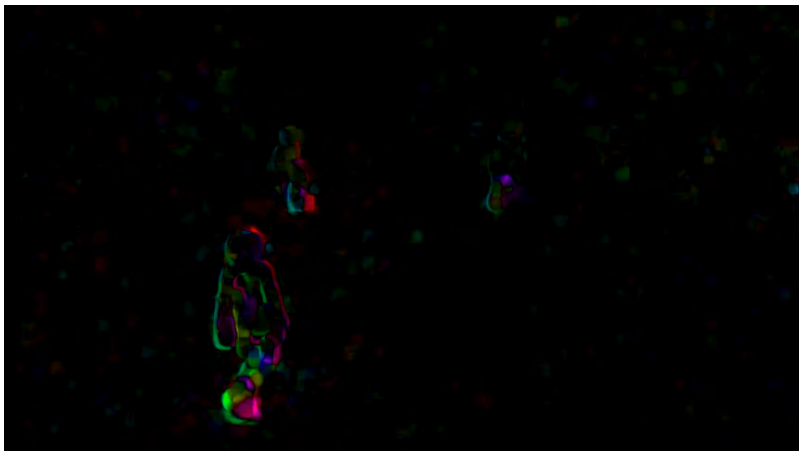

Figure 1.12 – Difference of Flow computed from three frames from the same scene as in Figure. 1.8

Similarly to Optical Flow, these methods focus on movement and discard some information about shape and color.

### 1.4.2 Audio

Audio is also a major modality when it comes to human behavior, as it carries most information about the voice and the sounds surrounding a person. Audio usually is preferred for communication over video only, and can as well be used at the same time as video. Our voice reveals a lot about our internal state. For example, sadness will tend to increase the pitch of the voice while anger will tend to lower it.

Thanks of our frequent usage of our voice, there are plenty situations where audio can be used to detect anomalous human behavior. Since audio also carries some information about someone's environment, it can also help us get more information about context which, as we have seen before, is mandatory for anomalous human behavior detection.

#### 1.4.2.1 Raw audio

Similarly to video, we can decide to use the raw audio stream to detect anomalous human behavior. Raw audio can be described by a 2-dimensional tensor (time, channels) where the number of channels typically is 1 or 2. However, because of its very high frequency (around $10^4$ Hz), audio can also have a high-dimensionality. Again, as with video, this motivated the development of several techniques to reduce the dimensionality of audio by extracting specific information, notably by focusing on human perception.

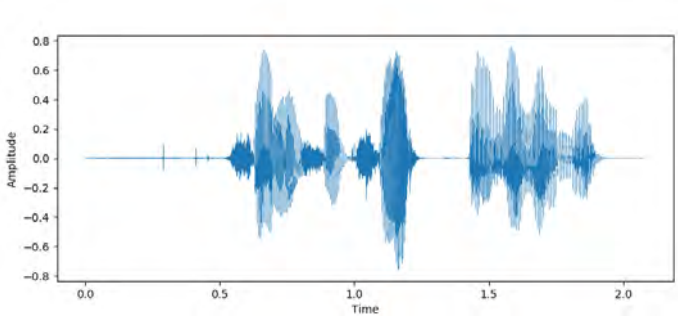


Figure 1.13 – Example of the waveform of speech with a frequency of 48 KHz and 2 seconds, for a total dimensionality of around $10^5$.

#### 1.4.2.2 Mel-Spectrogram

The waveform presented in Figure 1.13 does not provide us with a lot of information at first sight. Thanks to the amplitude of the signal, we can see that the sound is louder

at a few given moments, with a peak around 1.2sec. However, it is rather hard to see directly what is the frequency of the sound, or if we are dealing with a voice or something else.

Spectrograms will extract what are the different frequencies on a given time window, giving us the different estimated amplitudes for each frequency. This way, we can easily see if a specific frequency, and its harmonics, stands out, giving us the pitch of the sound.

Mel-Spectrograms are a kind of spectrogram designed to roughly match human perception, using the Mel(ody) scale. On this scale, higher frequencies seem to be closer than lower frequencies. In Figure 1.14, we show an illustration of a Mel-Spectrogram of the waveform presented in Figure 1.13. Thanks to this method, we can see this represents someone talking, as the dominant frequencies are characteristic of a voice.

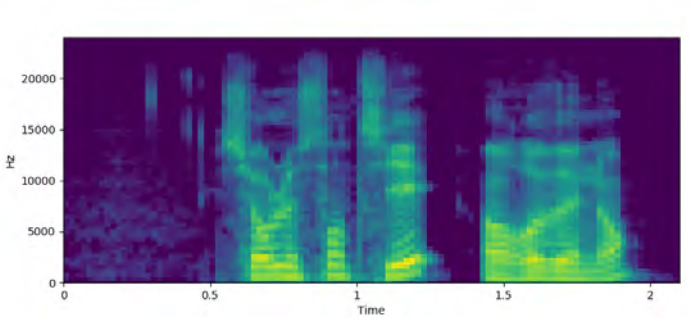


Figure 1.14 – Mel-Spectrogram based on the waveform shown in Figure 1.13. The voice can be more easily identified in spectrograms than in raw audio waveforms.

Moreover, these spectrograms can reduce the dimensionality of the audio. Similarly to raw audio, spectrograms can be represented by 2-dimensional tensors $(time_{spectrogram}, filters)$. However, spectrograms are computed by applying a strided convolution to the raw audio. The bigger the stride of the sliding window, the lower the $time_{spectrogram}$ will be, effectively reducing the dimensionality. Conversely, the more filters a spectrogram has, the bigger the dimensionality of the spectrogram. In practice, the dimensionality reduction because of the strided convolution is usually bigger than the increase because of the number of filters, meaning that spectrograms usually have a lower dimensionality than the raw audio they were computed from. In the example presented in Figure 1.14, the dimensionality was roughly divided by 10 (from $10^5$ to $10^4$).

#### 1.4.2.3 Speech content

With the emergence of Speech-to-Text tools, it is becoming feasible to extract words from an audio signal that contains speech. This is useful for anomalous human behavior detection in two cases:

1. What is expressed is anomalous in itself.
2. What is expressed does not match other elements of the voice, such as prosody.

### 1.4.3 Text

While we have seen text can be extracted from voice, it is not the only source of text. Specially with the popularity of social networks, humans express themselves more and more using text, making text a rather appealing modality for detecting anomalous behaviors in the framework of mental health monitoring for example.

### 1.4.4 Images

All modalities we have so far had a temporal/sequential component, but this is not strictly required when we talk about behaviors. It is also possible to identify behaviors on single images.

As with text, social networks come quickly to mind, as users regularly upload photos on them. These images can be analyzed relatively to:

1. Information coming with the image itself: textual description, timestamp, date of publication, etc...
2. The user who uploaded the image and their usual behavior.
3. The persons present on the image, if there are any, and their usual behavior (if known).

### 1.4.5 Body sensors

Human behaviors can also be reflected through various body signals that we can capture with body sensors. As we have talked before, there are a number of situations where anomalous behavior detection can help making diagnoses, e.g. neurodevelopmental disorders diagnoses when behaviors considered as normal are free of those disorders.

Body sensors have already been tested to detect such disorders or detect incoming extreme behaviors. Here are a few examples using body sensors displayed in Figure 1.15.

#### 1.4.5.1 Electrocardiography (ECG)

Bagirathan et al. 2021 used ECG to measure the heart rate of children to determine early indications of incoming emotional outbursts, resulting in *aggressive* and *extreme behaviors*.

#### 1.4.5.2 Electroencephalography (EEG)

Bosl, Tager-Flusberg, and Nelson 2018 used electroencephalography (EEG) to diagnose Autism Spectrum Disorder (ASD) on the basis of behavioral symptoms with *specificity, sensitivity [...] exceeding 95% at some ages*. Lau-Zhu, Lau, and McLoughlin 2019 also argue that *mobile EEG could open unprecedented possibilities for studying neurodevelopmental disorders*.

#### 1.4.5.3 Electrodermal activity (EDA)

EDA is used to measure changes in Skin Conductance Level (SCL). In their study, Prince et al. 2017 conclude that *participants with ASD had a greater change in SCL in response to all activities in the CSBS* (Communication and Symbolic Behavior Scale), showing that EDA can also be useful in revealing disorders.


Figure 1.15 – Examples of commonly used body sensors with potential for anomalous behavior detection.

It is worth noting that these body sensors still require well-controlled environments. For example, ECG is very sensible to patient movements (Sameni et al. 2007; Limaye

and Deshmukh 2016). This is a well-known limitation of those body sensors, which would currently restrict anomalous behavior detection to well-controlled environments.

Hopefully, these sensors are continuously being improved and may allow for a broader range of anomalous behavior detection tasks in the foreseeable future. For example, EEG classically did not allow for mobility. While there are still key challenges to solve when considering large-scale data collection, user comfort and reliability, great effort has been made to develop mobile EEG (Lau-Zhu, Lau, and McLoughlin 2019).

### 1.4.6 Network streams

Some part of network traffic is the direct result of human activity, allowing for a (limited) application of anomalous human behavior detection. A concrete example would be someone logging in and out at very similar times of the day as part of their routine, if they were to suddenly stop, we would flag it as anomalous considering the context.

Moreover, the notion of behavior can also be used when considering network streams. While this may not be as much related to humans, there is significant overlap between the methods one could use.

### 1.4.7 Online ratings

There are now plenty of websites where humans can give ratings and evaluations to a variety of things, such as hotels, movies, products, or even other humans.

These ratings can sometimes become overly important, as people will go as far as to hire people solely to give them positive reviews or give competitors negative reviews. Usually, such behaviors are not intended by the platform hosting the reviews. By considering faithful reviews as the norm, an anomalous human behavior detection system should flag existing and emerging behaviors leading to undesirable reviews.

In a similar way, some collaborative systems also use ratings to assess the importance of something. For example, the Tournesol project (Hoang 2021) is a growing collaborative effort to promote important, ethical, etc... videos in order to have a positive impact of society. However, the more this project will grow, the more malicious users it will attract. It will then be desirable to have an anomalous human behavior detection system to further improve the resilience of the project.

## 1.5 Multimodal anomaly detection in discourse using speech and facial expressions

The work presented in this document follows Cedric Fayet's thesis, which is called "Multimodal anomaly detection in discourse using speech and facial expressions" (Fayet 2018). In his thesis, Cedric Fayet worked on anomaly detection in discourse, a subset of anomalous human behavior detection. He focused on facial expressions and voice expressivity as they can carry information about a person's feelings and state of mind. Relatively to Cedric Fayet's thesis, we expand to all situation where human behavior can be observed, using modalities and methods these situations allow for.

More importantly to the present work, Cedric Fayet built Emo&ly, a multimodal dataset for anomaly detection in discourse. Emo&ly is particularly relevant for us as it is, to our knowledge, the first and only multimodal dataset for anomalous human behavior detection. We will go further in details in Section 2.2.1, where we talk about the different datasets at our disposal.

Chapter 2

# State of the Art

Now that we have introduced the subject and defined its delimitations (to the best of our ability), we can look into the state of the art (SOTA) related to our subject.

First, we will review methods that are commonly used in the SOTA. Then, we will see the main datasets used in the SOTA as well as some datasets that are of interest for our task. Finally, we will end this section with a talk about the recurring limitations found in SOTA methods and bring a critical point of view on this matter.

## 2.1 State-of-the-Art Methods

SOTA machine learning methods can usually be separated into two categories: methods that handle low-dimensional data, in low amounts, and methods that handle high-dimensional data, in high amounts. It is also possible for a method to be hybrid and borrow from both categories, usually trying to take advantage of the different strengths of both categories. However, as we will see, such methods can still be used in conjunction with a dimension reduction technique applied beforehand.

### 2.1.1 Methods suited for low amounts of data in low-dimensional spaces

These methods mainly refer to methods that were popular before the rise of Deep Learning, such as One-Class Support Vector Machines (OC-SVMs), Isolation Forests (iForests) and Local Outlier Factor (LOF). As the category suggests, they are usually better suited when samples have small amounts of features — i.e. data is present in low-dimensional spaces — and tend to generalize better on small amounts of data than the other category does. A common example of such data is the Iris Flower dataset (Fisher 1936), which contains 150 samples where each sample has 4 features (the length and width of the sepals and petals of the flowers).

#### 2.1.1.1 OC-SVM

First introduced in 1999 by Schölkopf et al. 1999, OC-SVMs define a hyperplane which is used to separate normal from anomalous data. OC-SVMs assume only a tiny percentage of presented data is anomalous — or almost anomalous — in order to define this hyperplane. Once defined, new data is then compared using this hyperplane: Samples that fall on the side of most normal data are considered as normal and samples that fall on the side of the tiny percentage are considered as anomalous.

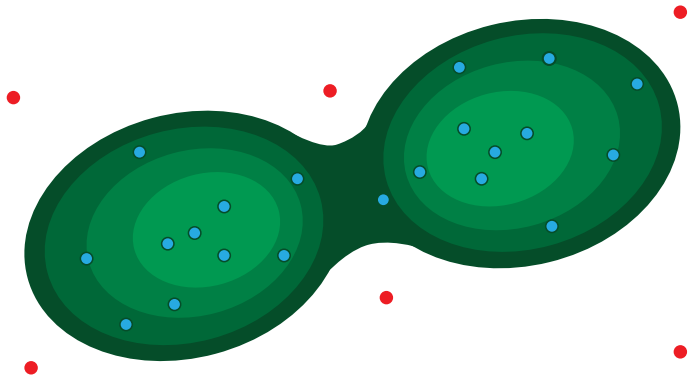

Figure 2.1 – Illustration of an OC-SVM used for anomaly detection. The green area depicts the side of the decision hyperplane where samples are considered normal. Samples outside the green area are therefore considered anomalous.

OC-SVMs have since then seen many improvements and publications in other fields of anomaly detection. Li et al. 2003 developed an intrusion detection method based on OC-SVMs to analyse logs. Y. Wang, Wong, and Miner 2004 extended kernel methods to the intrusion detection domain and combined them with OC-SVMs in order as well to detect intrusions and malicious attacks on computers and networks. R. Zhang et al. 2007 continued the work on OC-SVMs, this time to detect network anomalies. Chen, Qian, and Saligrama 2013 draw inspiration from both OC-SVMs — for their computational efficiency — and density-based methods — for their better performance —, yielding a method with lower complexity than density-based methods and better performance than OC-SVMs for anomaly detection.

In his thesis, which precedes the present one, Fayet 2018 achieved the best performance results on the Emo&ly dataset using OC-SVMs on audio only, video only and audio with

video. In all three cases, they used pre-computed features — such as formants for audio or facial landmarks for video — in order to reduce the dimensionality of inputs.

#### 2.1.1.2 iForests

iForests make similar assumptions: Anomalous data is rare and present characteristics that can be used to isolate them from the rest. In iForests, Isolation Trees (iTrees) — a form of binary trees — are built by randomly selecting one feature of the samples and a split value for this feature. In average, an anomaly will have a shorter path in the iTrees of the forest, as the nodes will in average isolate them earlier in a leaf. The length of the path can then be used to discriminate between anomalous and normal samples.

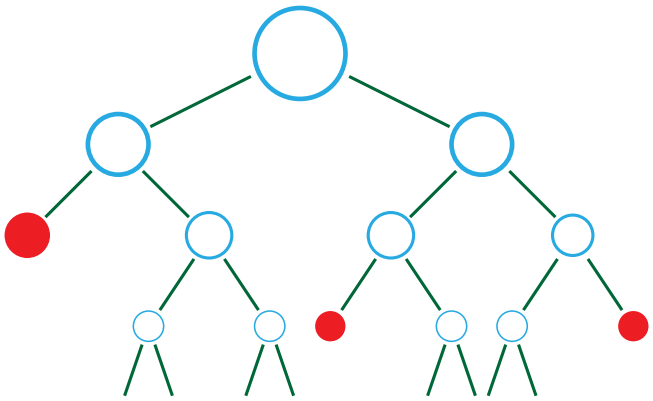

Figure 2.2 – Illustration of a single iTree from an iForest used for anomaly detection. In this example, red leaves are where samples are being isolated.

Both OC-SVMs and iForests can be fit using only normal data or with a tiny amount of anomalous data.

Recently, Marteau developed the HIF and DiFF-RF methods (Marteau, Soheily-Khah, and Béchet 2017; Marteau 2021), which are improvements over iForests. Using the DiFF-RF method, they were able to successfully detect anomalies and intrusions in 17 datasets, outperforming iForests in almost all cases, while challenging other methods such as the OC-SVM baseline.

However, in his thesis, Fayet 2018 tested iForests on the Emo&ly dataset. While still able to detect some anomalies, iForests yielded significantly worse results than OC-SVMs in all evaluated cases on the Emo&ly dataset.

#### 2.1.1.3 LOF

Breunig et al. 2000 developed the algorithm called LOF. This is a distance-based method, where the distance between a point and its $k$-nearest neighbors is used to compute an anomaly score for this point. The further away a point is from its $k$-nearest neighbors, the more likely it is to be anomalous.

As with OC-SVMs and iForests, LOFs are better suited for data laying in low dimensional spaces but not for the same reasons. Because high dimensional spaces are extremely sparse, a counter-intuitive phenomenon — referred to as the "curse of dimensionality" — occurs. As pointed by de De Vries, Chawla, and Houle 2010: For distance-based anomaly detection for high dimensional data, *the greatest challenge is how to deal with the 'curse of dimensionality': as the data dimension grows, distance measures lose their discriminative ability.* Because of this phenomenon, it would make little sense to compute distances directly in such spaces to detect anomalies.

For example, in the case of video surveillance, a few pixels turning black have the potential to create a bigger distance than any anomalous behavior could. Finally, as the LOF algorithm uses a $k$-nearest neighbors search, its computational cost scales quadratically with the number of samples, making this algorithm unsuitable for large amounts of data.

### 2.1.2 Methods suited for high amounts of data in high-dimensional spaces

Deep learning currently is the main method for handling high amounts of data in high-dimensional spaces — such as sound, images and videos — and we will focus on it during this thesis. In the framework of Anomaly Detection, autoencoders are by far the most popular class of deep learning methods. GANs (Goodfellow et al. 2014) also see some use. One of the strength of deep learning methods for anomaly detection is their ability to deal with high-dimensionality and their ability to scale with the amount of data.

Deep learning methods are also sometimes used to project data into a space with lower dimensionality, so they can be used by the previous category of methods.

It is worth noting that many variants exist for anomaly detection, for both autoencoders (Hasan et al. 2016; Sabokrou, Fathy, and Hoseini 2016; Luo, W. Liu, and Gao 2017; Chong and Tay 2017; Yousefi-Azar et al. 2017; W. Liu et al. 2018; Y. Zhao et al. 2017; Fan et al. 2018; Abati et al. 2019; Morais et al. 2019; Rodrigues et al. 2019; An and

Cho 2015) and GANs (W. Liu et al. 2018; Ravanbakhsh, Sangineto, et al. 2019), which tackle this problem in very different way. Since we cannot describe them all here, we only give the base principles of these methods.

#### 2.1.2.1 Autoencoders

The base principle of autoencoders is very simple: a model is taught to compress and decompress data, using an informational bottleneck.

First, an encoder encodes the input data, producing a compressed code. Then, the decoder decodes this code back to the input space, yielding a reconstruction of the input data. The error between the input data and the reconstruction is then back-propagated through the network in order to minimize the error.

As the training goes on, the model learns to produce compressed representations — mostly referred to as latent representations, latent codes or embeddings — and to minimize the error it makes when decompressing these representations.

When it comes to Anomaly Detection, it is assumed that autoencoders will produce a smaller reconstruction error for normal samples than for anomalous samples. Typically, a distance measure is used to compare the input data and the reconstruction, and yield single value which will be used to discriminate between anomalous and normal samples. Such distance measures often include Mean Absolute Error (MAE) and Mean Squared Error (MSE). The MAE is also known as the L1 distance, while the MAE is also known as the L2 or Euclidean distance. An illustration of an autoencoder being used for anomaly detection is given in Figure 2.3.

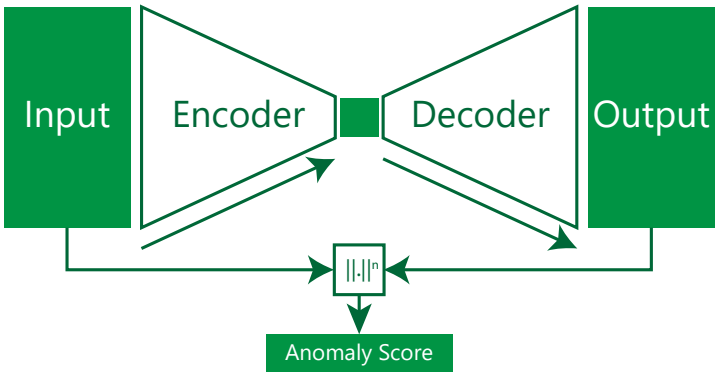


Figure 2.3 – A trained autoencoder used for anomaly detection. Samples are first encoded, yielding a latent representation which is lossy compression. The latent representation is then decoded, resulting in a lossy reconstruction of the input samples. The greater the difference between the input and the output, the greater the anomaly score and the more likely a sample is to be anomalous.

As discussed before, once trained, the Decoder can be discarded to only keep the Encoder which will produce low-dimensionality representations, which can then be used by other methods.

For example, Andrews, Morton, and Griffin 2016 successfully used latent representations from an autoencoder as the input features of a Radial Basis Function (RBF) $\nu$-Support Vector Machine (SVM) to detect anomalies. Additionally, they state they chose an autoencoder to have system that *would make no assumptions regarding [anomalies] other than their atypicality.*

In a similar manner, Erfani et al. 2016 developed an hybrid system where a Deep Belief Network (DBM) was used to *extract generic underlying features, and a one-class SVM is trained from the features learned by the DBM.* Their results showed that this method can yield similar performance to standalone autoencoders, while reducing the computational cost of training and testing by significant margins.

#### 2.1.2.2 GANs

Similarly to Autoencoders, introduced by Goodfellow et al. 2014, GANs were not first developed for anomaly detection but they can see use in this framework. GANs simulate a min-max game between two parts of the network: a generator and a discriminator. The

generator takes random noise and learns to produce a sample from it that the discriminator will take for a real sample. On its side, the discriminator learns to distinguish real samples from the ones the generator produces.

When the training is stable, this leads to a never-ending competition were both parts keep improving. In this case, we usually observe continuous improvement of samples produced by the generator. Once the training is done, the most common way to use GANs for anomaly detection is to ask the discriminator if it thinks the new sample is real or fake. The value yielded by the discriminator is then used to discriminate between anomalous and normal samples.

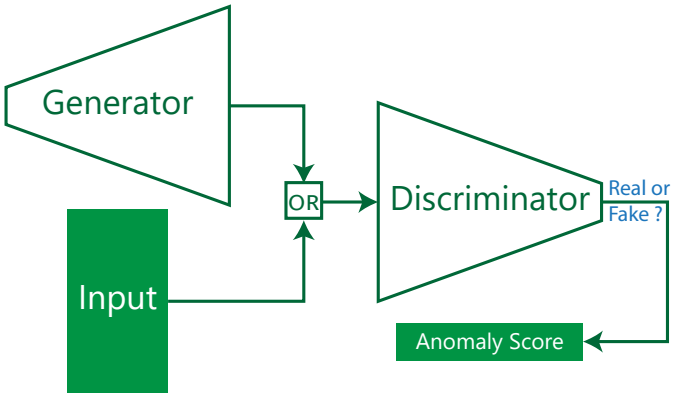


Figure 2.4 – Illustration of a GAN used for anomaly detection. The discriminator is trained to recognize forgeries from the generator, himself trained to fool the discriminator. For anomaly detection, the generator is discarded and the decision from the discriminator is used for determining if a sample is anomalous or not.

## 2.2 Datasets

As we have seen before, a sample (or a behavior) can only be considered anomalous according to a context. In all anomaly detection approaches, to our knowledge, the context is represented by a dataset. The model will usually train on this dataset and learn a boundary for normality from it.

We previously defined a context as a set of normal mechanisms that generate normal data. With sufficient data acquired, this set of normal mechanisms can be learnt by the model, approximating the context, thus enabling anomaly detection.

In this subsection, we will first present datasets directly related to anomalous human behavior detection. Then, we will present additional anomaly detection datasets that are not directly related to human behavior but can be useful to evaluate a method's ability to generalize outside the scope of anomalous human behavior detection. Finally, we will present miscellaneous datasets that can indirectly be used for improving anomalous human behavior detection, by performing tasks such as pre-training, features extraction, *etc.*

### 2.2.1 Anomalous behavior detection datasets

Here, we present 6 datasets for anomalous human behavior detection, where the first one is a multi-modal dataset targeting anomalies in the discourse and the 5 remaining datasets are video surveillance datasets.

#### 2.2.1.1 Emo&ly (Fayet et al. 2018)

Short for EMOtion and AnomaLY, Emo&ly is — to our knowledge — the first publicly available dataset for multi-modal anomalous human behavior detection. Indeed, as we will see with the rest of this section, most datasets available for anomalous human behavior detection are set in a specific framework: surveillance video. As this framework suggests, we only have access to one modality video as well (video).

Emo&ly fills this important void in publicly available resources. This corpus is composed of audio-video recordings of people reading a tale, directly facing camera. Each participant has been recorded reading the same tales 3 times, each time corresponding to a new situation:

1. Normal situation: They read the tale as they normally would.
2. Induced situation: While the participant is reading the tale, the experimenter tries to induce a reaction. Means include shifting or rotating the text, adding a funny background to the text, *etc.* In order to have as genuine reactions as possible, participants were not made aware of this beforehand.
3. Acted situation: Now that participant went through the second set of recordings, they are asked to act a given emotion while reading the tale.

Note that all situations were recorded at different occasions, leaving one week between recordings, so that the previous recording has minimal impact on their behavior.

Audio, video features and meta-data for the Emo&ly dataset are available on the Expression team website.

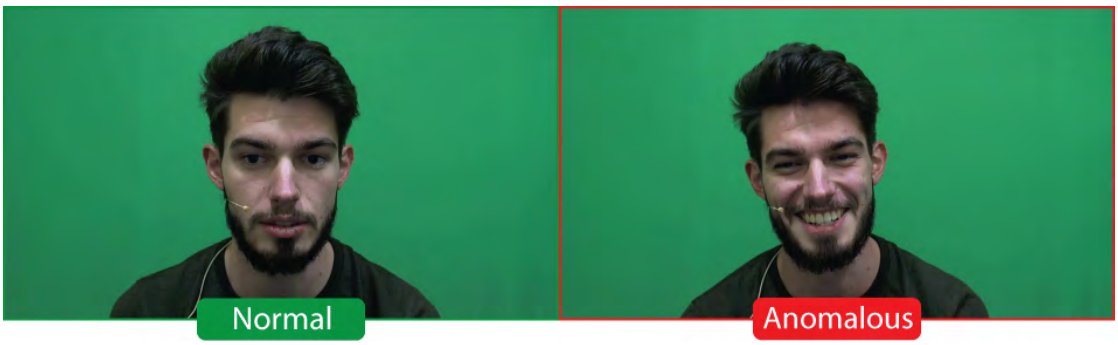


Figure 2.5 – Samples taken from the Emo&ly dataset. The two videos were recorded at different occasions, the samples displayed here were taken at the end of the same sentence from the same tale, but in the second one the anomalous behavior was induced.

#### 2.2.1.2 Video

The following datasets are the ones on which most experiments were conducted in the literature. The most studied sub-field of anomalous human behavior detection is the sub-field of surveillance video. Because of this, all presented datasets come from surveillance camera — or close to it — as we were unable to find any other publicly available dataset for anomalous human behavior detection besides these ones and Emo&ly.

**UCSD Anomaly Detection dataset (Mahadevan et al. 2010).** The University of California San Diego (UCSD) Anomaly Detection dataset (Mahadevan et al. 2010), most often referred to as the UCSD Pedestrian dataset, contains a set of grayscale videos taken from surveillance camera overlooking pedestrian walkways. Anomalous events are either the circulation of non-pedestrian objects (such as bikes and cars) or anomalous pedestrian motion patterns (such as walking on the grass). This dataset is separated in two parts, Ped1 and Ped2, corresponding to two different walkways and video resolutions.

UCSD Ped1 consists of 34 training videos and 36 testing videos. In addition to having frame-level labels for all 36 testing video, Ped1 also has pixel-level labels for 10 testing videos. All 70 videos have 200 frames. This dataset also contains a few corrupted frames.

UCSD Ped2 consists of 16 training videos and 12 testing videos. Ped2 has frame-level labels for all 12 testing videos. Each video has around 150 to 200 frames.

The UCSD Pedestrian dataset currently seems to be the most widely used dataset for video anomaly detection (Hasan et al. 2016; W. Liu et al. 2018; Luo, W. Liu, and Gao

2017; Fan et al. 2018; Ravanbakhsh, Nabi, et al. 2018; Ravanbakhsh, Sangineto, et al. 2019; Rodrigues et al. 2020; Hinami, Mei, and Satoh 2017; Y. Zhao et al. 2017; Chong and Tay 2017; Abati et al. 2019).

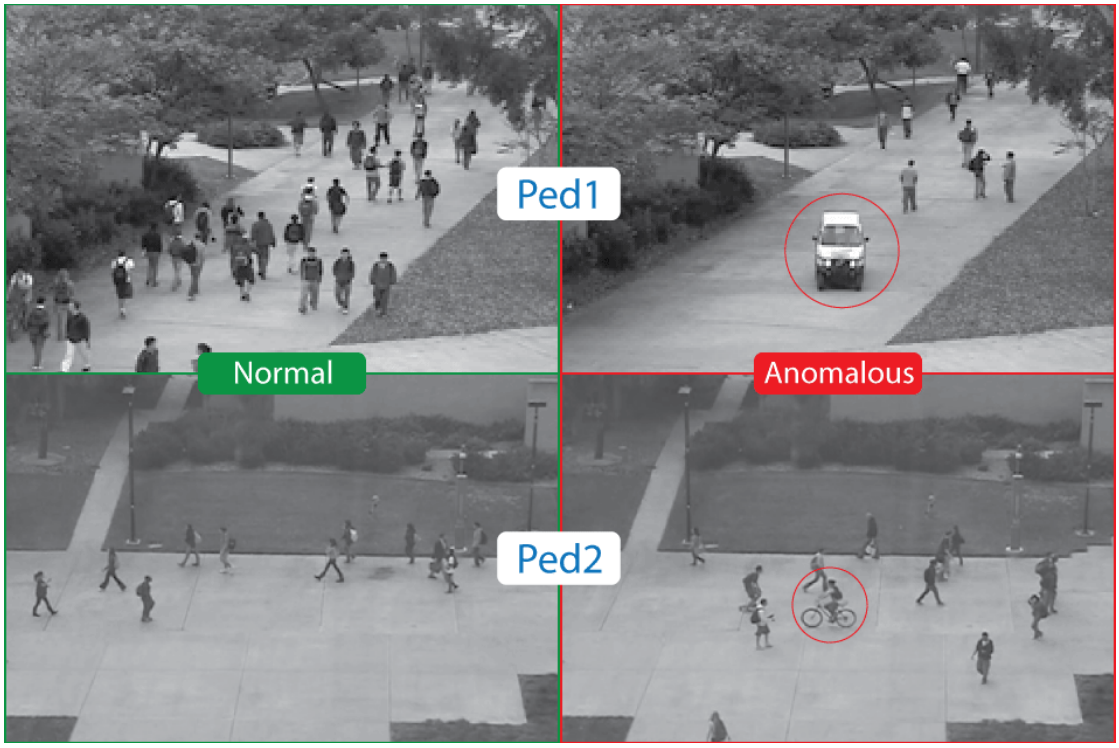


Figure 2.6 – Samples from both subsets of the UCSD Anomaly Detection dataset. Displayed anomalies in both cases are the presence of a vehicle.

The UCSD Pedestrian dataset can be downloaded from its dedicated page on the Statistical Visual Computing Lab (SVCL) website from the UCSD.

**Subway dataset (Adam et al. 2008).** Introduced by Adam et al. 2008, the Subway dataset contains five annotated videos from surveillance cameras. Authors consider anomalous events to be *all the unusual events which would interest a security guard watching these videos.*

Only two of them are situated in a subway, while the three remaining are set in a mall. Among the five videos, the two videos in the subway are the most commonly used in the literature and are often considered separately as Subway Entrance and Subway Exit.

The Subway Entrance is a 96 minutes long video where the 18 first minutes are free of anomalous events. The Subway Exit is a 43 minutes long video where the 10 first minutes are also free of anomalous events. Both videos are in grayscale. Anomalous events in these situations include experimenters going the wrong way — entering from the exit and vice versa — as well as trying to force their way in or out.


Figure 2.7 – Samples from the Subway Entrance and Exit subsets. Displayed anomalies in both cases are someone going in the wrong direction. On the top, an experimenter is trying to force their way out. On the bottom, the same experimenter is trying to force their way in, while people are exiting the subway.

The three mall video are 55, 40 and 60 minutes long video. Anomalous events mostly include people running in the mall, usually children or the experimenters. Contrary to Exit and Entrance, mall videos are in color.


Figure 2.8 – Samples from additional surveillance video footage from the Subway dataset. All additional situations happen in a mall. Displayed anomalies are people running.

To obtain the Subway dataset, please contact Amit Adam at the following mail address: email.amitadam@gmail.com.

**CUHK Avenue dataset (C. Lu, Shi, and Jia 2013).** The Avenue dataset (C. Lu, Shi, and Jia 2013), from the Chinese University of Hong Kong (CUHK), is made of 16 videos for training and 21 for testing, all videos are in color. In this dataset, the videos come from a fixed camera at human height. Anomalous behaviors were performed by the

experimenters and include events such as throwing objects, loitering and running.


Figure 2.9 – Samples from the CUHK Avenue dataset. The anomaly displayed here is someone throwing documents in the air.

The CUHK Avenue dataset can be downloaded from its dedicated page on the CUHK website.

**ShanghaiTech Campus dataset (Luo, W. Liu, and Gao 2017).** The ShanghaiTech Campus dataset (Luo, W. Liu, and Gao 2017) aims to fill a void present at the moment of its publication. Indeed, until now all datasets contain very few amounts of data, often are in grayscale, often have a poor resolution and often only contain one scene or two.

This dataset contains 330 training videos and 107 testing videos, all in color, from 13 different surveillance cameras. Authors tried to vary anomalous behaviors, which for example include events such as running, jumping, fighting or the presence of vehicles.

The ShanghaiTech Campus dataset can be downloaded from its dedicated page on the ShanghaiTech Vision and Intelligent Perception (SVIP) Lab github.io from the ShanghaiTech University.


Figure 2.10 – Samples from the ShanghaiTech Campus dataset. Only the video of 1 of the 13 surveillance camera is shown here but only the location changes, not the behaviors nor the anomalies. The anomaly displayed on the right is someone jumping in the middle of the way.

**ITTB-Corridor dataset (Rodrigues et al. 2020).** The ITTB-Corridor dataset (Rodrigues et al. 2020) brings similar value as the ShanghaiTech Campus dataset does.

This dataset contains 208 training videos and 150 testing videos, all in color, from a single scene and point of view. Authors included 12 types of anomalous behavior, where all behaviors result from group activity.


Figure 2.11 – Test samples from ITTB Corridor dataset. The anomaly displayed on the right is labelled as "Fighting".

The ITTB-Corridor dataset can be downloaded from its dedicated page.

**UMN crowd activity dataset (Mehran, Oyama, and Shah 2009).** The University of Minnesota (UMN) crowd activity dataset (Mehran, Oyama, and Shah 2009) contains 3 videos from overlooking surveillance cameras, for a total of 4 minutes of videos. One of the videos is in grayscale and the other two in color. All videos share the same pattern: At the start, people are randomly loitering in the designated space. At some point, a signal is given and all participants start running/fleeing the scene. Then, participants gather, loiter and a signal is again given, and so on. The anomalous segments are when the participants are running/fleeing.

While this dataset can be useful in a broad picture, it contains very little data and only one kind of anomalous events, making it unsuitable for experiments that would be performed only on one dataset.

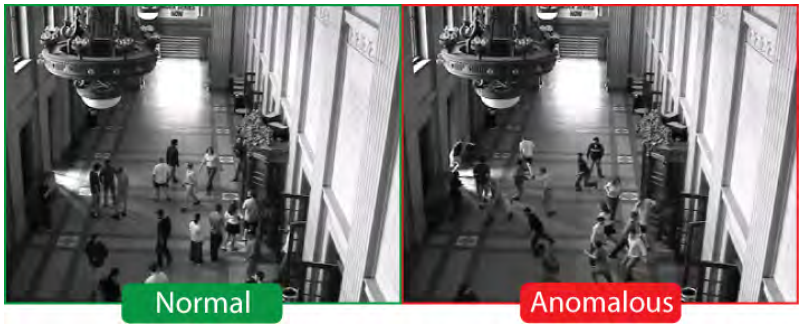


Figure 2.12 – Samples from one of the situations present in the UMN crowd activity dataset. People are loitering in the hallway waiting for a signal. When this signal is sent, people try to flee the hallway. The two frames displayed here were taken around a second apart.

The UMN crowd activity dataset can be downloaded from its dedicated page on the UMN website.

### 2.2.2 Additional anomaly detection datasets

Depending on the method used, testing on other anomaly detection datasets can be useful for ensuring methods that should generalize outside of human behavior actually do. We list here two datasets for anomaly detection, both on sequential data. The first one being an audio dataset and the second being a network intrusion detection dataset.

#### 2.2.2.1 Audio

Since audio is also a major vector of human expression and behavior, we include an audio dataset for anomaly detection besides Emo&ly. Unfortunately, this one is not directly related to anomalous human behavior detection.

**MIVIA road audio events dataset (Foggia et al. 2015).** The MIVIA road audio events dataset (Foggia et al. 2015) is composed of a total of 400 audio road surveillance events, where those events are either tire skidding or car crashes.

This dataset is not directly intended for anomaly detection but can still be used this way. In this case, anomalous events are also present in the training set in significant amount, but since those are labeled, they can be isolated from the training set.

Sounds are available in the form of audio files with durations around 1 minute, on which anomalous sounds (tire skidding and car crashes) were added to a normal road background sound.

The MIVIA road audio events dataset can be downloaded from its dedicated page on the MIVIA website. Note that downloading this dataset requires having or creating an account on the website.

#### 2.2.2.2 Network streams

Even if network streams do not necessarily convey information about human behavior, their sequential nature keeps us interested in them (for this thesis). In a sense, we can always approach network streams under a behavioral angle, where agents adopt a given behavior that will generate packets. Under this framework and in the context of Network Intrusion Detection (NID), malicious agents can be considered as having an anomalous behavior.

**Kitsune Network Attack dataset (Mirsky et al. 2018).** The Kitsune Network Attack Dataset (Mirsky et al. 2018) is a NID dataset that has been developed at Ben-Gurion University of the Negev, Israel. Records in the dataset represent IP packets which can be, for a part of them, sequentially dependent. The dataset is composed of 9 subsets covering 9 distinct attack situations, namely OS Scan, Fuzzing, Video Injection, ARP Man in the Middle, Active Wiretap, SSDP Flood, SYN DoS, SSL Renegotiation and Mirai Botnet.

In the framework of anomaly detection, network attacks are the anomalies we are trying to detect. At the exception of Mirai Botnet, all subsets have between $1.5 \cdot 10^6$ and $6.0 \cdot 10^6$ samples, where the first $10^6$ are free of anomalies and can be used for training models.

In our case, this dataset is very appealing because of the high number of samples it contains and the added diversity in modalities in our toolbox.

It is available for download on the UCI archive website or more simply on Kaggle.

#### 2.2.2.3 Toy datasets

It can also be useful to note that most datasets at least partially labeled, for anomaly detection or not, can be used to test anomaly detection methods. In order to do that, all one has to do is to define a rule of what is normal depending on these labels. For example, one could take the "Hello World" of Machine Learning, the MNIST dataset, and remove a given number from the training set and consider instances of this number as anomalies.

However, keep in mind the resulting dataset can potentially be very artificial and, because its anomalies would resemble each others a lot, they may be too easy to detect. A method should never be solely evaluated on such datasets.

### 2.2.3 Miscellaneous datasets

#### 2.2.3.1 Audio-video

**Audioset (Gemmeke et al. 2017) .** The Audioset dataset (Gemmeke et al. 2017) is a large collection of video segments with sounds, where each segment comes from a YouTube video and has been labeled by hand. The labels given are not mutually exclusive, some even include others, for example a video labeled "Female Speech" will also be labeled "Speech".

The dataset references a total of $2 \cdot 10^6$ video segments and each segment is about 10 seconds long, yielding a of total $5.5 \cdot 10^3$ hours of audio-video.

In our case, we are mostly interested in human behavior, so we will focus on videos with the label "Speech". About half of the videos have this label, which represents $10^6$ video segments for a total of near 2.8 hours.


Figure 2.13 – Four examples from the Audioset dataset from diverse situations. This diversity is important in order to model as much behaviors as possible and have a better model of each behavior. It is also helpful to increase robustness. However, some situations will be particularly difficult like the one presented in the top-right corner, where the sound and the video are not well aligned. Situations like the one in the bottom-left corner are not uncommon in Audioset and are helpful for modeling behaviors mostly observed through audio.

While Audioset cannot really be used to evaluate an anomalous human behavior detection model, is can be useful for building representations of human behaviors. Additionally, having videos with audio, Audioset is multi-modal dataset. This can be useful for building joint representations of audio and video by identifying the mutual information between those modalities.

We will discuss more in Section 5.1.1 about how datasets such as Audioset can be useful in our case.

#### 2.2.3.2 Action video datasets

**Kinetics (Smaira et al. 2020).** The Kinetics dataset (Smaira et al. 2020) is meant for Action Recognition. It exists under several versions: 400, 600 and 700, where the number indicates the number of classes (actions) present in the dataset.

In a similar way to the Audioset dataset, the Kinetics dataset is a collection of video segments of about 10 seconds from YouTube. However, not all videos from the Kinetics dataset have audio, making it less useful for multi-modal modeling.

In our case, we are not really interested in the labels provided with the dataset, as we are more likely to use this dataset in a similar manner to Audioset. That is to say, extract representation for human behaviors. However, as described in the previous section, the labels can still be useful to produce a toy dataset for anomaly detection, where samples from a few given classes are excluded from the dataset.

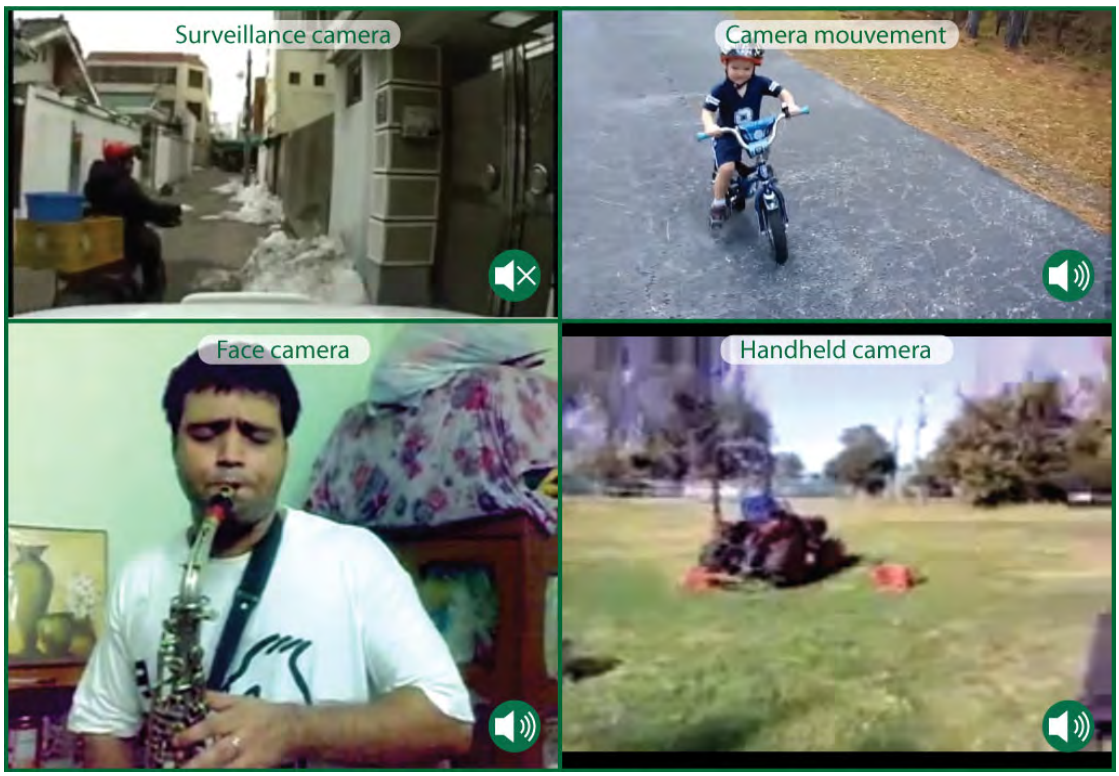


Figure 2.14 – Four examples from the Kinetics dataset from diverse situations. Contrary to the Audioset dataset, sound is not often useful or even present. In situations like those displayed in the right corners, most of the sound comes from behind the camera. In situations like the one in the top-left corner, almost no sound can be heard.

#### 2.2.3.3 Online ratings

**Tournesol (Hoang 2021).** In Section 1.4.7, we talked about the Tournesol project (Hoang 2021), a collaborative effort for promoting videos based on user ratings. Contrary to other platforms where videos are being rated, Tournesol focuses on ethics and moral. Because of this, robustness to malicious intents is of the utmost importance to ensure a group of people cannot attack the system in order to promote unethical videos, for example.

Anomaly detection, and more specifically anomalous human behavior detection, seems to be a valid approach for detecting such attacks. The Tournesol project is open-source and the public part of its database can be downloaded. It consists in entries about the ratings made by users and include features such as the values of the rating, a timestamp,

the user ID, *etc.*

While there is currently no Tournesol dataset for anomaly detection, it may come in the future, as the project is still young and interested in anomaly detection. Moreover, modeling behaviors from the data available is a new challenge, raising new questions such as: What features should we include in a sample ? How important is the temporal context ? (*i.e.* ratings made by other users in the same time frame) How relevant is a user's history ? And much more questions. This has the potential of yielding new insight on modeling human behaviors, which can be useful for broader anomalous human behavior detection.

## 2.3 State-of-the-Art Limitations

Before starting to develop new methods, it is necessary to assess what is currently lacking in the literature. We identify two major types of limitations that are unrelated to the method used and are often, if not always, present in the literature: Dataset limitations and experimental limitations.

As we will see, these limitations are to be corrected before deploying anomalous human behavior detection systems in the real world, to avoid situations like overfitting or too many false positives.

Overall, the work presented in this thesis was made with most of these limitations in mind in order to provide a solid support for future works.

### 2.3.1 Dataset limitations

#### 2.3.1.1 The amount of data available

The datasets we listed in Section 2.2.1 share a similar problem: they do not contain a lot of data, ranging from a few minutes to a few hours of video. Even in the case of methods better suited for low amounts of data, this is problematic for two reasons. First, the dimensionality of each sample is very high and with a small amount of samples, the support of the underlying distribution of the dataset becomes extremely small.

The most representative example of this phenomenon probably is the UCSD Pedestrian Dataset, where Ped1 and Ped2 only have 34 and 16 samples respectively, for a total of 6800 and 3060 frames for training, where each frame has an initial dimensionality of 37604 (238 by 158 by 1) and 86400 (360 by 240 by 1) respectively. As we can see from

the comparison of this "frames per dimension ratio" between different video datasets in Table 2.1, the number of points is extremely sparse.

Note that frames with higher resolutions can reasonably be downsampled by a factor up to 50. The dimensionality can also be reduced a manageable amount by extracting and only keeping low-dimensionality data such as keypoints (facial landmarks, body joints, ...). However, such process may discard relevant information for anomaly detection and more importantly, a method based on human pose does not transfer to videos such as face-camera videos, where facial landmarks would be more relevant.

| Dataset | Total time | Videos | Frames | Frame dim. | Ratio |
|---|---|---|---|---|---|
| Emo&ly | 03:19:42 | 451 | 299550 | 6220800 | 0.048 |
| UCSD Ped 1 | 00:11:20 | 34 | 6800 | 37604 | 0.180 |
| UCSD Ped 2 | 00:05:06 | 16 | 3060 | 86400 | 0.035 |
| Subway Entrance | 00:18:00 | 1 | 27000 | 196608 | 0.137 |
| Subway Exit | 00:10:00 | 1 | 15000 | 196608 | 0.076 |
| CUHK Avenue | 00:10:04 | 16 | 15100 | 691200 | 0.022 |
| ShanghaiTech | 03:07:31 | 330 | 270024 | 1232640 | 0.219 |
| ITTB-Corridor | 05:22:37 | 358 | 483924 | 1232640 | 0.393 |
| Kitsune subset | N/A | N/A | 1000000 | 115 | 8695.7 |

Table 2.1 – Comparison of different datasets for video anomaly detection. The ratio is computed from the total number of training frames in a dataset divided by the dimensionality of a single frame, higher is better. In this regard, the ShanghaiTech and ITTB-Corridor datasets are significant improvements. The NID dataset Kitsune is also indicated for reference.

This problem is magnified by the fact, as said previously, UCSD Ped2 is by far the most used dataset in the literature on the topic of surveillance video anomaly detection. Previous works usually reach very high scores on the UCSD Pedestrian Dataset, with little guarantee the model will generalize on new data.

This is the second reason why small amount of data available is problematic: it becomes easier to overfit a method or its parameter in order to improve results by a small margin. One might observe a given pattern in the dataset and unconsciously build a method around that, assuming this pattern will always be valid. For example, none of the listed dataset in Section 2.2.1 contains rain. However, rain can significantly impact human behavior and computer vision algorithms, but in a real world situation, this change in behavior

is expected and no one would ever want to trigger an alarm for that case. If no method is ever tested on video with rain, how is one supposed to deploy an anomalous human behavior detector in the real world, with the assurance that the system will not fail as soon as rain begins to fall ?

Rain is only one example of the many things absent from these datasets, night time is another rather common situation which is never covered in listed datasets. In the case of rain specifically, not only we could see rain on the image, but it would also affect the area and the behavior observed, as people usually move faster to avoid the rain. However, since rain is completely missing from these datasets, we cannot better estimate the impact it would have.

Ideally, the amount of gathered data should be several orders of magnitude greater, and should be gathered with the goal of covering more context in mind. This should help cover more of these unforeseen situations where a behavior would *obviously* be normal but would be flagged as anomalous by the system.

The low amount of data creates an additional problem: the context defining normality is too narrow. As we have seen before, the normality of a sample is considered relatively to a context, which is itself represented by the dataset. If not enough samples are gathered from the context, we will probably end up with a incomplete definition of normality.

#### 2.3.1.2 Annotations and a fuzzy definition of normality

The Emo&ly and the Subway datasets also share another problem: their definition of an anomalous behavior is fuzzy. This results in annotators having to give subjective labels: one might think a given behavior fits the normal rule while another one might not. This could potentially be interesting if annotators were numerous but this is unlikely to happen.

**Emo&ly dataset (Fayet et al. 2018).** Labels currently available for distinguishing between normal and anomalous segments are often lacking or are inaccurate. This is mostly because of two reasons: 1) Annotators did not have time to fully process the dataset and a vast part of it remains to be annotated. 2) The annotators are asked *to annotate reactions that are unexpected during the reading of the slide [...].* The fuzzy nature of these instructions very likely contributed to the impreciseness of the resulting labels.

Additionally, there are not as much anomalies as one would have expected. Participants

did not necessarily react to the stimuli intended to induce an anomalous behavior. During the acted situation, participant did not often let themselves be caught up in the game and read the tale in the same way as before.

In spite of those limitations, Emo&ly remains the only corpus of its kind, which is why it is still very desirable to work with this dataset at the moment. It is worth noting that, while currently halted, the annotation process of the Emo&ly dataset may resume in the future.

**Subway dataset (Adam et al. 2008).** As with Emo&ly, the definition of anomalous events for the Subway dataset is rather fuzzy, leading again to imprecise labels. In the case of Subway Entrance, several publications use different labels for this dataset and it is not always clear which labels are used. For example, multiple works seem to use labels introduced by J. Kim and Grauman 2009, as they report the same number of anomalous events (66, up from 20).

We find labels introduced by J. Kim and Grauman 2009 to be too loose, as even simple loitering at the top of the stairs is labeled as anomalous, even if this would likely not be of interest for a security guard. As shown in Figure 2.15, such event happens at least once in the dataset and would likely be considered anomalous in a real-world setup.

However, people loitering before the gates are close to the camera, and thus impact more pixels on the image. For methods depending on the number of pixels affected by an event — such as the one presented by J. Kim and Grauman 2009 — people loitering are more likely to trigger an alarm. This observation may explain why those events were included in the list of anomalous events.


Figure 2.15 – In this example, a girl is waiting for someone else for more than one minute. The definition given by J. Kim and Grauman 2009 includes *people [who] loiter for a long time*. According to this definition, this event should raise an alarm.

#### 2.3.1.3 The absence of a validation set

Most, if not all, anomaly detection dataset are split in two: train and test sets. This contrasts with other domains such as classification, where a third part is usually present: the validation set. The absence of validation sets in current datasets most likely is due to the low amount of data, to avoid a further reduction of the sizes of the train and test sets. The lack of a validation set also contributes to publishing overfit models.

### 2.3.2 Methodological shortcomings

Now that we have reviewed the main problems we identified with the datasets commonly used in the literature, we will go over some frequent methodological shortcomings in previous works.

#### 2.3.2.1 Low dataset count per publication

We have taken a significant number of publications and looked at which datasets experiments were conducted on in these publications. As we can see from Table 2.2, only a few publications perform tests on more than 3 datasets, with an average of $\sim 2.6$ datasets per publication. When reading this table, it is worth noting that Ped1 and Ped2 are very much alike and should probably be considered as the same dataset, this is also valid

for Entrance and Exit. Some works in this list were published before the ShanghaiTech dataset was introduced, showing the popularity of this new dataset is relatively high. However, the UCSD Pedestrian dataset remains king, probably to serve as a common ground between works, despite its shortcomings.

| Publication | Ped1 | Ped2 | Entrance | Exit | Avenue | ShanghaiTech |
|---|---|---|---|---|---|---|
| Hasan et al. 2016 | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Hinami, Mei, and Satoh 2017 | | ✓ | | | | |
| Y. Zhao et al. 2017 | ✓ | ✓ | | | ✓ | |
| Chong and Tay 2017 | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Feng, Yuan, and X. Lu 2017 | ✓ | | | | ✓ | |
| Luo, W. Liu, and Gao 2017 | | ✓ | | | ✓ | ✓ |
| Sabokrou, Fayyaz, et al. 2018 | ✓ | | | ✓ | | |
| W. Liu et al. 2018 | ✓ | ✓ | | | ✓ | ✓ |
| Fan et al. 2018 | ✓ | ✓ | | | | |
| Ravanbakhsh, Nabi, et al. 2018 | ✓ | ✓ | | | | |
| Abati et al. 2019 | | ✓ | | | | ✓ |
| Ravanbakhsh, Sangineto, et al. 2019 | ✓ | ✓ | | | | |
| Rodrigues et al. 2020 | | ✓ | | | ✓ | ✓ |
| Morais et al. 2019 | | | | | | ✓ |

Table 2.2 – Datasets on which publications conducted anomaly detection experiments.

The low dataset count per publication is a problem because it gives little guarantee the methods described in these publications can generalize outside of the datasets they use, and were not overfit on a small selection of datasets. Moreover, as discussed in 2.3.1, with the exception of the ShanghaiTech dataset, all of these datasets are particularly small, increasing the chances for overfitting to occur.

This problem motivates us to test the methods developed during this thesis on as much datasets as possible, while varying the hyper-parameters between datasets as little as possible. This way, our experiments should tend to produce more robust methods, which we find more desirable than surpassing state-of-the-art methods on a single dataset.

#### 2.3.2.2 Low reproducibility

Looking at the same publications as in the previous section, we can also see there is a major reproducibility problem. As we can see from Table 2.3, out of 14 publications, only

4 provide their code and out of these 4, only 2 provide pre-trained parameters for their respective models. None of the 10 publications with no code provide their pseudo-code which is essential in anomaly detection. The pseudo-code usually includes procedures such as pre-processing, the way the anomaly score is computed, and so on...

| Publication | Pseudo-code | Code | Dataset | Model |
|---|---|---|---|---|
| Hasan et al. 2016 | | ✓ | ✓ | |
| Hinami, Mei, and Satoh 2017 | | | ✓ | |
| Y. Zhao et al. 2017 | | | ✓ | |
| Chong and Tay 2017 | | | ✓ | |
| Feng, Yuan, and X. Lu 2017 | | | ✓ | |
| Luo, W. Liu, and Gao 2017 | | ✓ | ✓ | |
| Sabokrou, Fayyaz, et al. 2018 | | | ✓ | |
| W. Liu et al. 2018 | | ✓ | ✓ | ✓ |
| Fan et al. 2018 | | | ✓ | |
| Ravanbakhsh, Nabi, et al. 2018 | | | ✓ | |
| Abati et al. 2019 | | ✓ | ✓ | ✓ |
| Ravanbakhsh, Sangineto, et al. 2019 | | | ✓ | |
| Rodrigues et al. 2020 | | | ✓ | |
| Morais et al. 2019 | | | ✓ | |

Table 2.3 – Table to measure the level of reproducibility of previous works, depending on the availability of: the method's algorithm, the code used to perform experiments, the data and the final model (i.e the parameters and hyper-parameters of a model after fitting).

This problem is not really surprising, as the reproducibility crisis seems to affect all domains of science, but this motivates us to provide ways to fully reproduce our work in this thesis.

#### 2.3.2.3 Lack of statistical analysis of performance

The vast majority of deep learning methods and a lot of other methods for anomaly detection are of stochastic nature. The results of these methods can sometimes greatly vary depending on the random draws made during execution (e.g. random initialisation, dataset sampling, latent distribution sampling, ...). It is thus desirable to make sure the

method did not only work because of a lucky random seed, but rather it works in the general case.

Ideally, stochastic methods should be run around 10 times, using a different random seed each time, and then report the performance score's mean and variance across these 10 runs. However, in practice it is very rarely the case. In particular, deep learning methods sometimes require days or weeks to be trained, making it unpractical to run 10 times the same training procedure. For example, out of the 14 works cited earlier, none of them perform such analysis.

While it should usually affect the performance results of a method, it is still useful to keep this shortcoming in mind when evaluating the overall confidence in a method. In the case of this thesis, we are focusing on deep learning methods and did not perform such statistical analysis.

#### 2.3.2.4 Lack of performance metrics

As we will see later in Section 3.3, there are several ways to measure a system's performance on a anomaly detection task. All these performance metrics are more or less relevant depending on the subsequent tasks the method could be used for. There is not a "best" performance metric but previous works usually only report 1 or 2 of these metrics, as we can see in Table 2.4.

The reported metrics are usually the same, which is useful for comparing methods between them. This way, it becomes easier to elect a "winner" and "beat" state-of-the-art methods. However, reality is a bit more subtle, as methods have different strengths and weaknesses. Evaluating a method with more performance metrics often reveals that, if a method did not improve on the usual metric, it may have improved on another metric (and vice-versa).

Reporting only 1 or 2 performance metrics is useful for the publication process, where reviewers can focus on seeing if the new method beats the previous one on 1 metric. But if the new method beats the previous one on 2 out of 4 metrics, the choice of which is better becomes harder, and this choice usually plays a big factor in deciding if a paper should be published or not, instead of judging a method on its own merit. However, for someone who would want to deploy an anomaly detection system in the real world, having more than 2 performance metrics is essential, as they will be able to select the one(s) that fit their problem and their situation.

| Publication | Metrics count |
|---|---|
| Hasan et al. 2016 | 2 |
| Hinami, Mei, and Satoh 2017 | 2 |
| Y. Zhao et al. 2017 | 2 |
| Chong and Tay 2017 | 2 |
| Feng, Yuan, and X. Lu 2017 | 2 |
| Luo, W. Liu, and Gao 2017 | 1 |
| Sabokrou, Fayyaz, et al. 2018 | 2 |
| W. Liu et al. 2018 | 1 |
| Fan et al. 2018 | 2 |
| Ravanbakhsh, Nabi, et al. 2018 | 2 |
| Abati et al. 2019 | 1 |
| Ravanbakhsh, Sangineto, et al. 2019 | 2 |
| Rodrigues et al. 2020 | 1 |
| Morais et al. 2019 | 1 |

Table 2.4 – Number of performance metrics used to evaluate methods per paper. In all cases, the authors provided the value for the Area under the Receiver Operating Characteristic curve (usually called AUC/ROC). In the case where authors reported 2 metrics, the second is always the Equal Error Rate, which is also computed from the ROC curve.

Having considered this lack, we will be using at least 3 performances metrics when evaluating the methods developed in this thesis. We will detail the performance metrics we use to evaluate our work in Section 3.3.

One small note on false positives: The false positive rate is generally taken into account when considering the performances of an anomaly detection system. However, the weight given to false positives should depend on the context and the final user. For example, in the frame of a security application, the final user may accept a greater number of false positives in order to reduce the amount of false negatives.

### 2.3.3 Recurring limitations of state-of-the-art models

We finish our review of the current limitations of the SOTA with some elements that are commonly missing in models. These elements usually are missing by design or because of a lack of literature.

#### 2.3.3.1 Concept drift and Active learning

Concept drift is a notion that warns us about the fact that the real world can and will change over time. This is particularly important in anomaly detection. In Section 1.2.1, we insisted on the notion of context in anomaly detection and on the fact that a sample can only be deemed anomalous relatively to a context.

As the real world will change, so will the different contexts that are part of this world. A striking real-world example of such change is the drift that happened when considering same-gender relationships. As Brewer 2014 shows, support for gay marriage has significantly evolved these last 50 years, changing from being seen as anomalous by the general population to becoming supported in several countries such as the United Kingdom, the United States of America, Canada and France.

While situations such as the previous example usually unfold on long periods of time, there are also situations where the context changes on relatively short periods. For example, this is the case for web-based software systems and general online activity, where a user's preferences and habits may evolve. While concept drift in anomaly detection system has been well studied (M. Ma et al. 2018; Zambon, Alippi, and Livi 2018; Saurav et al. 2018), methods to adapt to it are completely missing from most works on anomaly detection. In particular, none of the works on anomalous behavior detection we listed in Tables 2.2, 2.4 and 2.3 accounts for concept drift.

Note that Active learning can be a solution to help adapting to concept drift in some situations, such as video surveillance. As the concept drifts, the number of errors of an anomaly detection system increases, giving more opportunities for an operator to give feedback to the model, effectively adapting the model to the drift.

However, the works we listed in the previous paragraph do not perform AAD either. Overall, relatively to anomalous human behavior detection and to the best of our knowledge, there exists no work either accounting for concept drift or performing AAD.

#### 2.3.3.2 Lack of human activity detection

To our knowledge, very few work include a human activity detection module. Methods which rely on human pose estimation (Morais et al. 2019; Rodrigues et al. 2020) usually perform this step, as they usually first crop the image around humans before estimating the pose. As we have seen with datasets, the real world is far more complex than what these datasets represent, and numerous situations could trigger a false alarm. Among

those situations, there are some where no human activity would be observable, so we should at least make sure there is some human activity before "detecting" an anomalous behavior. Since human activity detectors can also produce false positives and negatives, they would likely change the performance results of all the methods that did not include them, for better or for worse.

#### 2.3.3.3 Multimodal anomalous behavior detection

They have been extensive work both on multimodal anomaly detection (Park, Erickson, et al. 2016; Park, H. Kim, and Kemp 2019; Nedelkoski, Cardoso, and Kao 2019; Park, Hoshi, and Kemp 2018) and on multimodal modeling of data more or less related to human behavior (Hu et al. 2020; Owens, Jiajun Wu, et al. 2018; Y. Zhou et al. 2018; Owens and Efros 2018). However, to our knowledge, the only work tackling the challenge of multimodal anomalous behavior detection was made by Fayet 2018 during his thesis. And even there, the different modalities were not modeled jointly to detect anomalies. Instead, they were modeled independently and only then the resulting features were aggregated.

Overall, joint multimodal modeling is a hard task, mostly because the modalities usually lie in high dimensional spaces. The computational cost of joint multimodal modeling becomes quickly prohibitive when considering cases such as audiovisual data: the amount of training data required increases, the training time increases, the size of the model increases, *etc.*

Because joint multimodal modeling is a difficult and not fully solved challenge in itself, it is not surprising to not have seen it be tackled at the same time as anomalous human behavior detecting.

Thankfully, this leaves plenty of room for future works and innovations on the subject of multimodal anomalous behavior detection.

Chapter 3

# Methodology

Before we go into the details of our contributions, we introduce three notions commonly used in the literature and that we used in our contributions. To be more precise, we present instances of these notions.

First, we present four different learning paradigms and their interest relative to this thesis. In second, we will describe the different classes of models we studied during this thesis. And third, we will talk about the different tools and metrics used for evaluating anomaly detection systems.

## 3.1 Learning Paradigms

Machine learning models for anomaly detection must be trained on data before we evaluate them or before they are put in production. Data allow models to learn how to separate anomalies from normal samples. There exist several learning paradigms we can use to train our models. These learning paradigms define which parts of the data the model will see during training, and consequently, the heuristics we can use for training our model.

In this subsection, we will describe four different learning paradigms, namely supervised learning, unsupervised learning, semi-supervised learning and self-supervised learning.

### 3.1.1 Supervised learning

The first learning paradigm we start with is supervised learning. In supervised learning, all samples of a dataset $\mathcal{T}$ have a matching label, forming pairs in such manner that for a dataset, we have $\mathcal{T} = \{(x_i, y_i)\}_{i=1...N}$ where $x_i \in \mathcal{X}$ are the sample and $y_i \in \mathcal{Y}$ their matching labels. Therefore, in supervised learning, we have $\forall x_i \exists y_i$.

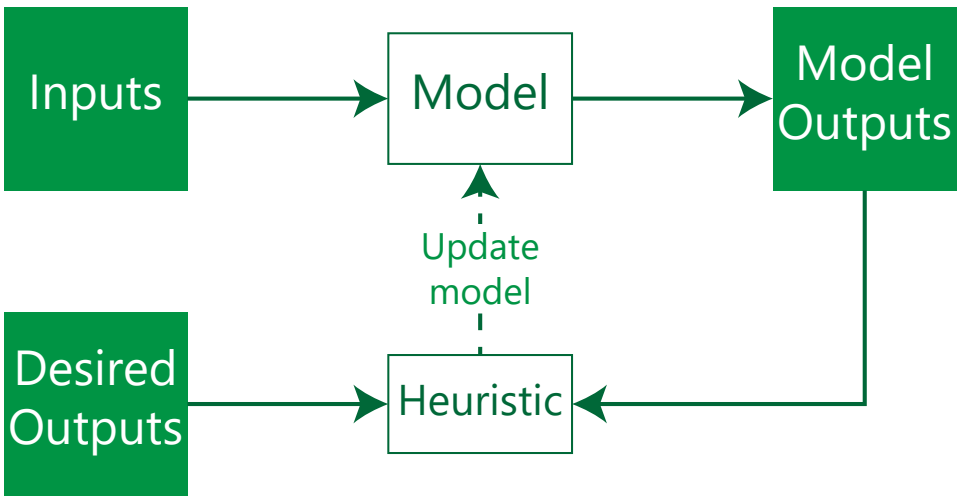


Figure 3.1 – When using Supervised Learning, the model first infers outputs from the inputs it receives. Then, the model outputs are compared to the desired outputs, the "good answers", by means of a heuristic (such as categorical cross-entropy). The model is finally updated in order to optimize the heuristic. This cycle repeats itself from there until the model is considered fit.

However, supervised learning can scale poorly as the number of samples grows. Each sample has to be manually labeled by a human, which takes time and money. This is particularly problematic for anomalous human behavior detection, as the data concerned usually take more time and effort to annotate due to their sequential nature.

A typical application of supervised learning is image classification, where a given image is associated with one or more classes (humans, vehicles, animals, ...). Digit recognition on the MNIST dataset (LeCun, Léon Bottou, et al. 1998) is a common introductory example, where images of handwritten digits must be associated with the digits written on it.

Nonetheless, supervised learning hardly works with anomaly detection. In addition to the low scalability of the labeling procedure, anomaly detection is not about modeling two classes where the first one would contain normal samples and the second one would contain anomalies. Indeed, as said before, anomalies are unpredictable and thus cannot be modeled. Instead, we only want to model normality. Additionally, anomalies are, by definition, much rarer than normal samples. This is commonly referred to as the imbalanced classification problem and results in models predicting classes in minority less often than they should.

Overall, these problems explain why supervised learning is almost never used in the anomaly detection literature. It is however useful to understand supervised learning to understand the following learning paradigms.

### 3.1.2 Unsupervised learning

As the name suggests, what differentiates unsupervised learning from supervised learning is the fact that $\mathcal{Y}$ cannot be directly accessed. While unsupervised learning removes the constraint of having to label every sample, this does not solve the problem of imbalanced data and more importantly, requires different heuristics / objectives than those used in supervised learning.

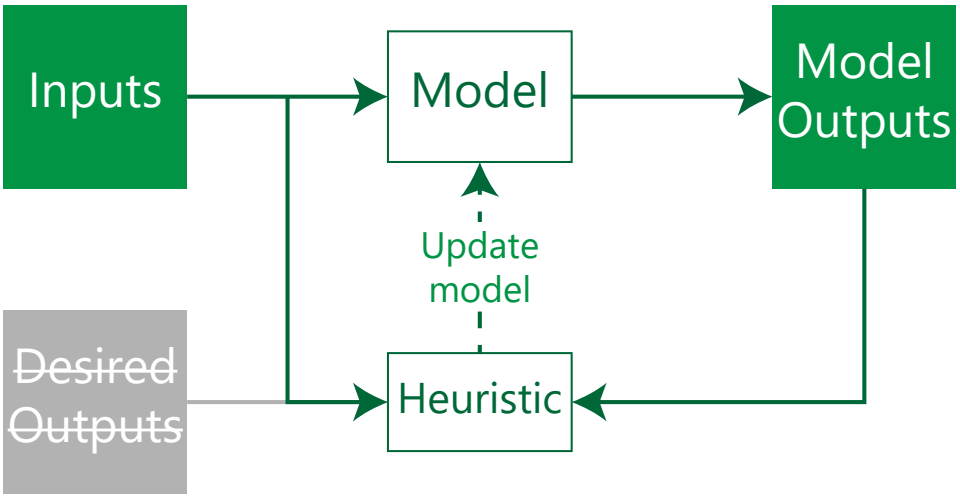


Figure 3.2 – Compared to supervised learning, unsupervised learning does not require labels and inputs are used instead for computing the heuristic.

Clustering and autoencoding are two common applications of unsupervised learning, both relevant to anomaly detection.

Put simply, in clustering, we do not know the classes that our dataset could be made of, but we assume they exist. We then try to organize our data around a set number of centroids, yielding regions of data points. In the case of anomaly detection, samples that do not belong to said regions are considered anomalous.

As previously explained in Section 2.1.2, in autoencoding the goal is to learn to compress data and then decompress it while producing as few errors as possible while re-

constructing data. In the case of anomaly detection, the model will learn to efficiently compress common data and will produce more errors on uncommon data. Therefore, samples having more errors than a set threshold will be considered so uncommon as to be anomalous.

However, unsupervised learning is also rare in the anomaly detection literature and is often mistaken for semi-supervised learning, the next learning paradigm we will talk about. This is due to how datasets for anomaly detection are built, as anomalies are most of the time removed from the training dataset.

### 3.1.3 Semi-supervised learning

Semi-supervised learning usually refers to the combination of supervised and unsupervised learning, in the sense that only a small portion of the dataset is labeled. Said otherwise, only a small part of $\mathcal{Y}$ can be directly accessed.

In the frame of anomaly detection, the semi-supervised approach refers to the absence of anomalies in the training dataset: $\forall y_i \in \mathcal{Y}_{training}, y_i = 0$, where 0 refers to the label "normal". This absence of anomalies in the training dataset makes it possible to only learn from normal data and thus only model normality.

This is how all datasets presented in Sections 2.2.1 and 2.2.2 are built. To our knowledge, this is the case for the vast majority of anomaly detection datasets, hence why most anomaly detection methods would fall under the paradigm of semi-supervised learning.

This small difference between semi-supervised learning and unsupervised learning has two consequences we need to take into account.

First, the cost of labeling of supervised learning is re-introduced but at a lower scale: we only need to make sure no anomalies are present in the training dataset. Additionally, anomalies identified this way can be put in the test dataset if such dataset is planned. However, even if labeling for semi-supervised learning can be quicker than for supervised learning, it can still become a major issue. For example, as the Emo&ly dataset is made of about 1300 videos lasting between 10 and and 70 seconds, includes audio and contains subtle anomalies, it is only partially labeled despite the hours of work put in.

Second, the absence of anomalies in the training dataset means we have little to no guarantee the method will be resilient to the presence of a low amount of anomalies in another dataset.

From the perspective of a potential user of an anomaly detection methods evaluated under the paradigm of semi-supervised learning, this means that they have to make sure

no anomalies are present in their data when training their model. This also means they should plan how they are going to acquire data, such that as little effort of labeling has to be made (e.g. only acquiring video from situations known to be normal).

In order to follow the literature and be able to compare our methods with previous works, we will only be using the semi-supervised learning paradigm in this thesis in order to train anomaly detection models.

### 3.1.4 Self-Supervised learning

Self-supervised learning is a learning paradigm that recently gained in popularity due to its ability to help learn better latent representations and the broadness of its applications.

Self-supervised learning can be viewed as a subset of unsupervised paradigm where the goal is to exploit the order found in data. For example, given two adjacent sections of an image, what is their spatial relation ? It is the first one directly above the second one ? On its right ? Another example shown in Figure 3.3 involves distorting the input data and trains the model to produce similar representations for the distorted and non-distorted views of the same sample, while separating from representation of different samples.

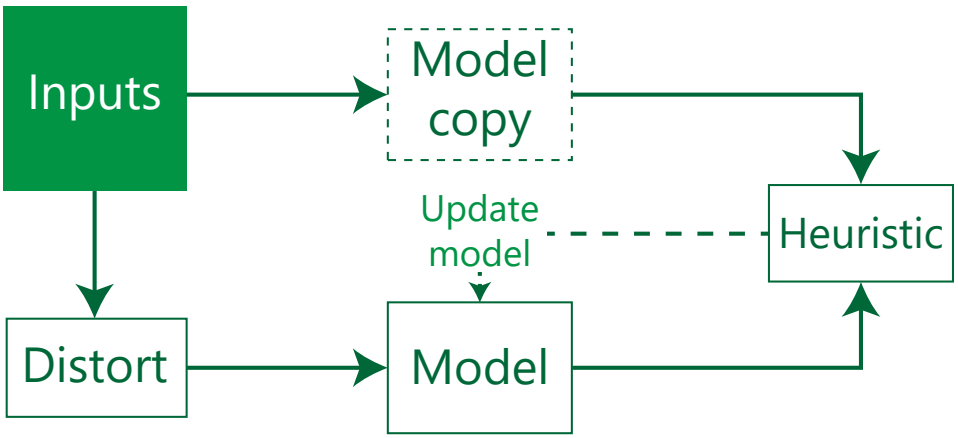


Figure 3.3 – In this example of self-supervised learning, only one of the two inputs is being distorted. Both versions of the input go through the same model and the heuristic is chosen to encourage the model to produce similar outputs/representations for both versions of the same input. The heuristic must also make sure the model does not collapse by mapping all inputs to a single point.

While not directly useful for anomaly detection, self-supervised learning is essential to representation learning, on which we will expand further in Sections 5.2 and 6.6. Finally, we note that self-supervised learning is not mutually exclusive with either unsupervised or semi-supervised learning.

## 3.2 Classes of Models

There exists a rich and vast collection of models in the literature of machine learning we can use for anomaly detection. In this thesis, we focus on deep learning models. In this section, we will describe the different classes of models we studied in this thesis. It is worth noting that a model can belong to several of these classes, as they are not mutually exclusive. For each of these classes, we will explain how they work and, more importantly, why we are interested in them in the context of this thesis. A review of these classes of models will also be helpful in understanding subsequent sections, the main results section in particular.

### 3.2.1 Autoencoders

In Section 2.1.2, we previously addressed the basics of autoencoders in the framework of anomaly detection. In Figure 3.4, we present a concrete example where an autoencoder is used to detect anomalies on the UCSD Pedestrian dataset (Mahadevan et al. 2010).

We will highlight three variants of autoencoders: Predictors, Variational autoencoders and U-Nets.

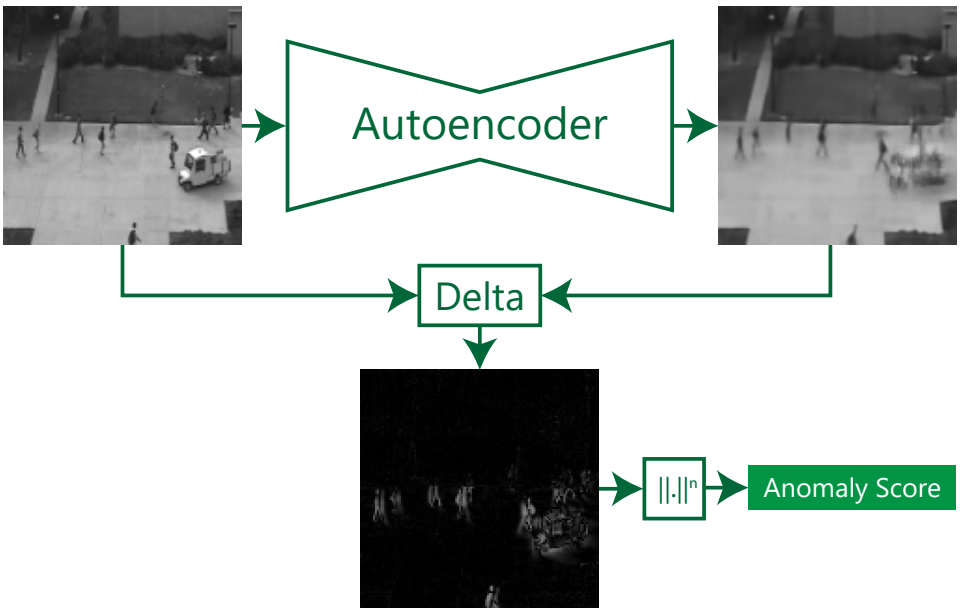


Figure 3.4 – In this example, an autoencoder was trained to reconstruct normal samples from the UCSD Pedestrian 2 dataset. During anomaly detection, it fails to reconstruct the vehicle, yielding a greater error as shown in the bottom image. The norm of this error is then used to compute an anomaly score: a greater error indicating a greater probability of having an anomaly and vice-versa.

#### 3.2.1.1 Predictors

First introduced by Y. Zhao et al. 2017 under the name Spatio-Temporal Autoencoder (STAE), predictors are autoencoders where a second decoder has been added to the autoencoder. Predictors take a sequence as input, the first encoder is tasked to reconstruct this sequence while the second decoder is tasked to predict what comes after the sequence (typically, a second sequence of the same length as the input sequence). We illustrate this class of models in Figure 3.5.

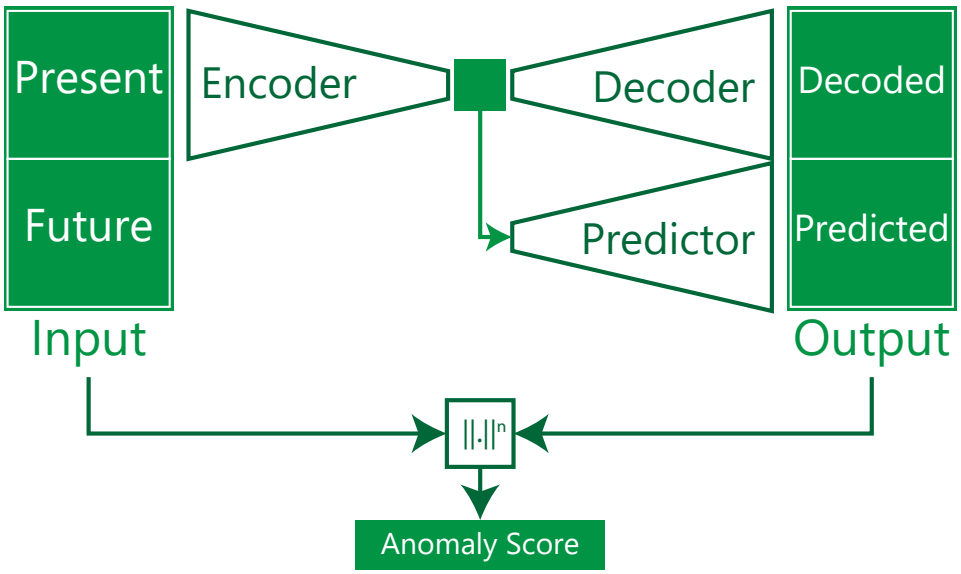


Figure 3.5 – With predictors, the input in split into parts, referred to as the present and the future. We then task the model to reconstruct the present and predict the future based on the present. Since in this case we know both the present and its future, we can compute the error as we do with autoencoders, allowing us to train the model then compute an anomaly score.

The second decoder is mainly useful during training, as it encourages the encoder to produce latent representations from which the future can be hypothesised. Afterwards, the second decoder may or may not be discarded for detecting anomalies. If the second decoder is discarded, the predictor becomes a simple autoencoder during anomaly detection. Otherwise, the error of the second decoder is added to the error of the first one. As predicting the immediate future is easier than predicting the distant future, Y. Zhao et al. 2017 suggest to add a decreasing weight to the error of the second decoder, where the error of the first frame of the future has a weight close to one, while the last frame has a weight close to zero.

It is worth noting that, because of the way they are trained, predictors learn in a self-supervised manner.

Overall, predictors are suitable to improve the modeling of temporal sequences, which is particularly relevant to us as we are studying human behavior.

#### 3.2.1.2 Variational Autoencoders

First introduced by Kingma and Welling 2013, Variational Autoencoders (VAEs) are a probabilistic extension of autoencoders. In basic autoencoders, the encoder directly outputs a latent representation, which is then fed to the decoder. In VAEs, the encoder outputs a latent distribution instead, typically a normal distribution. This distribution is then sampled to produce the latent representation.

As shown in Figure 3.6, the encoder usually outputs two tensors with the same shape as the expected latent representation, the first tensor acting as the mean of the normal distribution and the second tensor as the variance — or the log of the variance — of the normal distribution.

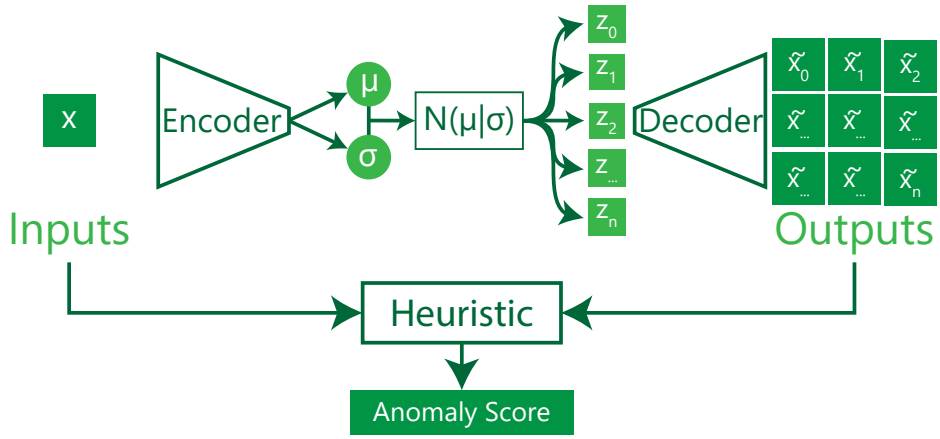


Figure 3.6 – Contrarily to basic autoencoders, the encoder in VAEs produces a latent distribution instead of a single latent representation. From this distribution, we can sample multiple latent representations that can then be decoded, yielding multiple reconstructions. Having access to multiple outputs gives us the opportunity to use new heuristics. For example, we can use the lowest reconstruction error among all outputs to only consider the most likely output, or we can use the mean reconstruction error as An and Cho 2015 did.

While it was not intended for anomaly detection at first, VAEs are interesting in this framework. For example, if we consider the predictors we have seen previously, we can say it is not really possible to predict the future with certainty. Instead, we can predict multiple futures with different probabilities of happening. In their current state, predictors can only predict the most likely future at best, or a weighted mean of all likely futures.

The probabilistic aspect of VAEs can help alleviate this problem by allowing the model to make as many predictions as we sample the latent distribution. Assuming we observed a likely future, but not the most likely one, by sampling this latent representation multiple times, we give the opportunity for the model to predict this observed future. In this case, an anomaly would be a sample that is so unlikely that the model would be unlikely to predict.

In practice, An and Cho 2015 successfully used VAEs to detect anomalies, by computing a Monte Carlo estimate of the reconstruction error, sampling the latent distribution multiple times.

As we have highlighted in Section 1.3, one of the specificities when working on anomalous human behavior detection is the uncertainty surrounding said behaviors. VAEs give us new tools for modeling this uncertainty.

#### 3.2.1.3 U-Nets

First introduced by Ronneberger, Fischer, and Brox 2015 for biomedical image segmentation, U-Nets also are similar to autoencoders. The main difference comes from the addition of skip connections at different levels of the model as shown in Figure 3.7, creating wider and wider bottlenecks until they are completely by-passed by the top-level skip connection, which directly links the input to the output.

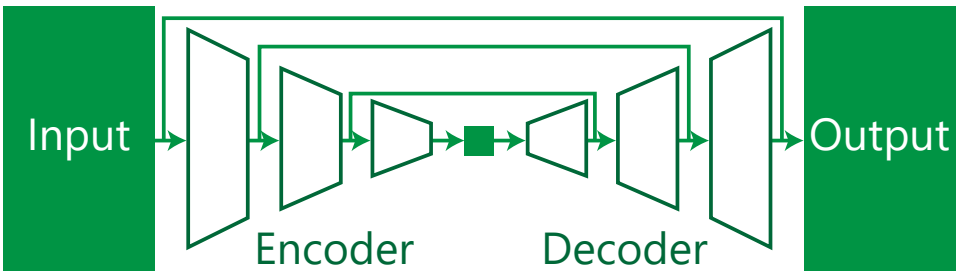


Figure 3.7 – An illustration of a U-Net with four stages and a skip connection for each stage. These stages include the usual bottleneck of autoencoders, two intermediate stages and the top-level stage, directly connecting the input to the output.

If the task of U-Nets were to reconstruct the input, it would just have to let the input take the top-level skip connection directly to the output, making the task trivial. However, U-Nets are usually not asked to reconstruct the input, but to produce a different view of

the input in the same input space. For example, in the paper they were introduced by Ronneberger, Fischer, and Brox 2015, U-Nets are used to transform biomedical images into their respective segmentation map.

In the framework of anomaly detection on sequences, U-Nets can be asked to output the future, in a similar way to what predictors do, as if the first decoder was completely discarded. For example, in their work on video anomaly detection, W. Liu et al. 2018 use a U-Net to predict a single frame in the future from a video.

As with predictors, this potential for improving temporal sequences modeling is particularly interesting when considering human behaviors.

### 3.2.2 Adversarial models

As for autoencoders, we previously introduced adversarial models in Section 2.1.2, or rather an instance of adversarial models called GANs. While GANs are generative models, they can be of use for other tasks such as anomaly detection. The application of GANs we saw in Section 2.1.2 discarded the generator and only used the discriminator to detect anomalies but there exist other ways to detect anomalies using GANs and their variants.

We previously mentioned the work of W. Liu et al. 2018 in Section 3.2.1 when we talked about U-Nets. Their model also contains a discriminator during training, as shown in Figure 3.8. This discriminator is used *for generating a more realistic [future] frame*. We note that the addition of a discriminator can also be helpful to prevent the U-Net from simply copying the last frame from the present, as this is a trivial solution for the U-Net which would yield good performance using the MSE loss. Only the U-Net is kept after training and anomaly scores are derived from the difference from the predicted future frame and the actual future frame, in the same way autoencoders are used for anomaly detection.

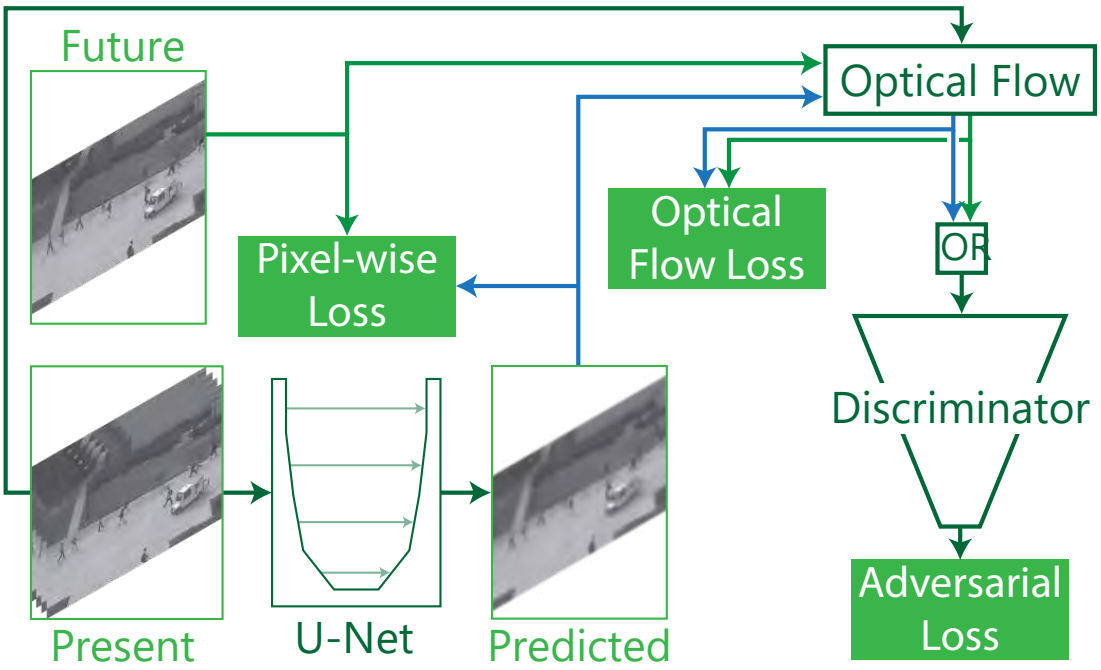


Figure 3.8 – An illustration of the training process of the method introduced by W. Liu et al. 2018. There are 3 objectives for the U-Net. First, it is trained to predict the next frame (pixel-wise loss). Second, the optical flow between the present and the predicted frame must match the expected optical flow (optical flow loss). Finally, the same resulting optical flow should be able to fool the discriminator (adversarial loss).

While not meant for anomaly detection, we also note the work of Larsen et al. 2016 on VAE-GANs. As the name suggests, VAE-GANs are a combination of VAEs and GANs, where the decoder of the VAE and the generator of the GAN are the same model, as shown in Figure 3.9. VAE-GANs also introduce something new that is of interest to us in the context of anomaly detection: a new similarity metric. Most autoencoders use the MSE as their similarity metric to both train the model and compute anomaly scores. As we will see later in Section 5.2.1, MSE and other similarity metrics in the input space have some major flaws.

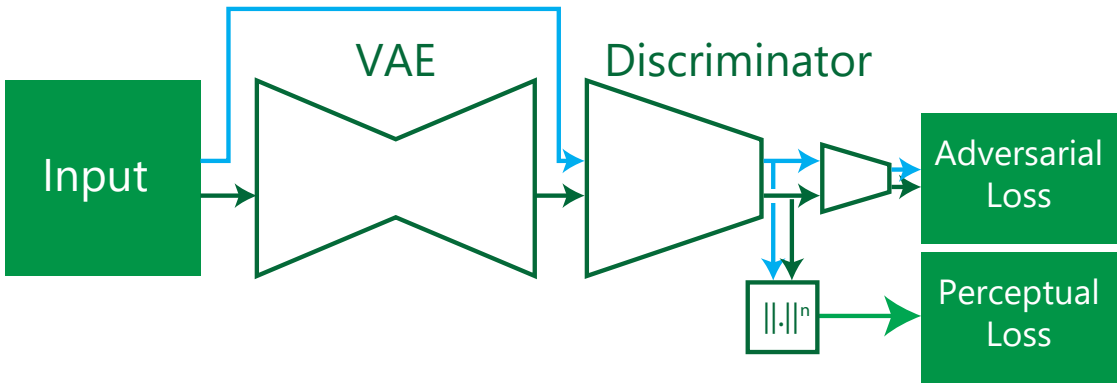


Figure 3.9 – An illustration of a VAE-GAN during training. As with GANs, the discriminator is trained to separate real examples from generated examples while the VAE is trained to fool the discriminator: This is the adversarial loss. However, the VAE is also trained to reconstruct the input in a high-level space, which is allowed by the discriminator: This is the perceptual loss.

On the other hand, the similarity metric introduced in VAE-GANs does not happen in the input space. Instead, VAE-GANs use the discriminator by passing the input through the first $l$ layers, yielding an intermediate representation. This intermediate representation is then compared using a simple distance measure such as the MSE. The addition of a discriminator allows the usage of this perceptual loss and can replace the MSE for both training the model and computing anomaly scores.

Overall, adversarial models also give us new tools to work with. First, they give us access to a new metric, in the form of the discriminator's output, which can be used for computing the anomaly score of a sample. Second, they can be used to improve other models, as W. Liu et al. 2018 did in their work. Finally, they can give us access to new metrics. These metrics, such as the perceptual loss of VAE-GANs (Larsen et al. 2016), are helpful for both training our models and detecting anomalies.

### 3.2.3 Autoregressive models

When studying human behavior, the data we use are mostly temporal sequences. This motivates us to investigate the potential usage of autoregressive models, a class of models meant for sequential data.

Autoregressive models try to answer the following question: given a sequence of pre-

vious observation, what is the most likely element to come next ? As shown in Figure 3.10, autoregressive models are used for forecasting sequences. Most of the time, these sequences are time series (Vaswani et al. 2017; Oord et al. 2016; Rushe and Mac Namee 2019; Radford et al. 2019), but not always as images can be seen as sequences (Van Oord, Kalchbrenner, and Kavukcuoglu 2016; Dosovitskiy, Beyer, et al. 2020) and even features of latent representation can be used in a sequential manner (Abati et al. 2019).

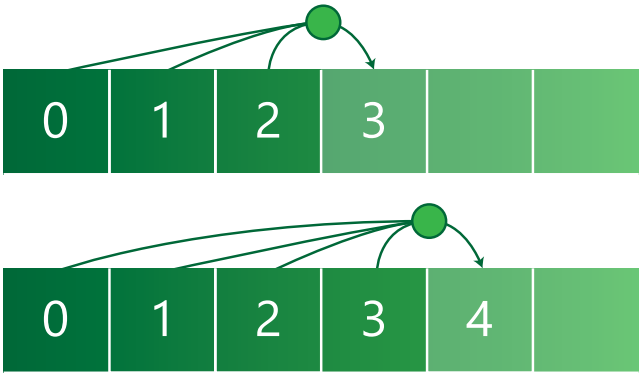


Figure 3.10 – Illustration of two steps from an autoregressive model. The model has access to the previous elements of a sequence and predicts the next element of this sequence. In this example, the model starts with access to the three previous elements and gives a prediction about the fourth element. This new predicted element is then added to the sequence, so the prediction about the fifth element can be done, *etc.*

Moreover, most deep autoregressive models — if not all — are probabilistic models. They do not directly produce the next element of the sequence, but rather a probability distribution on the different elements possible conditioned on previous elements. This allows us to model uncertainty in a similar way as to VAEs, which interest us for the same reason we gave for VAEs: human behavior is highly uncertain by nature.

Overall, autoregressive models interest us for anomalous human behavior detection for these two reasons: forecasting behaviors while considering uncertainty. Given that anomalies are unpredictable, autoregressive seem like good candidates.

#### 3.2.3.1 Transformers

Since their introduction by Vaswani et al. 2017, transformers have seen tremendous success. Originally designed for Natural Language Processing (NLP) and translation,

transformers have been improved on continuously since then (Choromanski et al. 2020; Beltagy, Peters, and Cohan 2020) and used for new tasks such as image classification (Dosovitskiy, Beyer, et al. 2020).

Transformers are mostly built on the idea of attention. Attention is a mechanism that allows a model to give more or less weight to some of its inputs. An illustration of attention used in a simple autoregressive model is given in Figure 3.11.

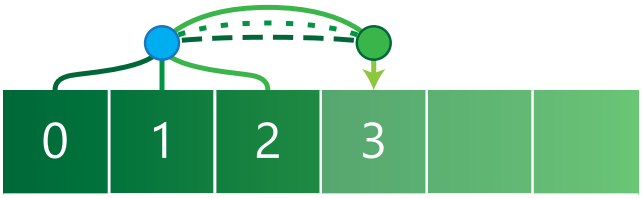


Figure 3.11 – Before predicting the next element of the sequence based on previous elements (the green circle), attention is first computed from these elements (blue circle). This gives more or less weight to each element, as illustrated by the length of the dashes in the lines after the attention mechanism. In this example, the bottom line has a greater weight than the middle one but a smaller weight than the top one.

As illustrated in Figure 3.12, the base transformers are made of an encoder and a decoder. As transformers were built for translation, the encoder was designed to handle the source language, while the decoder was made to handle the target language. Both parts perform attention on their respective inputs. As the encoder only has access to the source language, it uses attention on it: this is called self-attention. On his side, the decoder first also uses self-attention on the previous words it translated. After that, the decoder uses attention on the output of the encoder based on the result of its own self-attention and the output of the encoder.

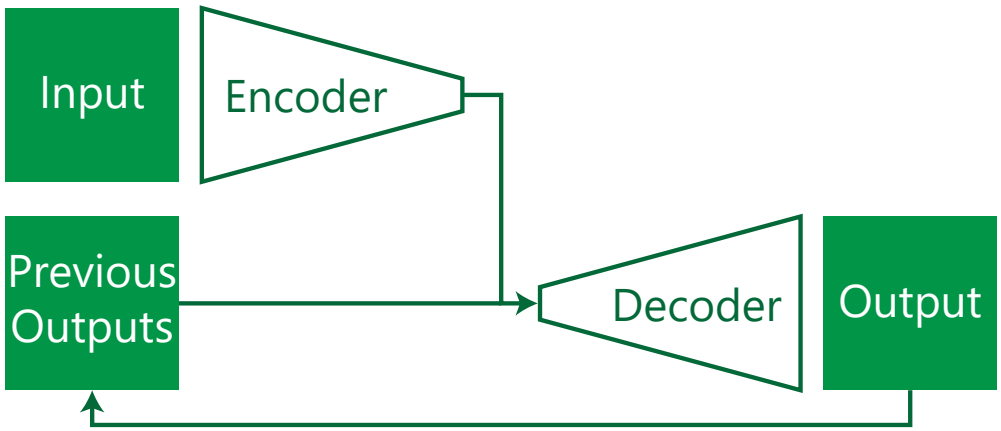


Figure 3.12 – The original transformer is made of two parts: an encoder and a decoder. The decoder is tasked to output elements in the target domain and is conditioned by both its own previous outputs and the output of the encoder. On his side, the encoder handles the source domain.

While transformers were invented for translation, they can be used almost as-is for other tasks that imply domain transfer on sequences. Additionally, the fact that the decoder of the transformers has access to the previous elements it produced and uses them to produce the next element makes it an autoregressive model. This gives the decoder the ability to become a generative model: For example, in the GPT-2 model (Radford et al. 2019), only the decoder was kept and can be used in a manner closer to the one illustrated in Figure 3.10.

Overall, transformers interest us, with or without their encoder but for different reasons. If we consider the decoder-only version, transformers can be used as described above: for forecasting human behaviors based on the present. Their encoder-decoder counterpart can be used in the same way, where the encoder would only handle the present the decoder the future. Here, the domain transfer would be from the present to the future. Since we are also interested in using multiple modalities, the encoder could handle one modality while the decoder handles another one. For example, the encoder could receive audio while the decoder is trying to produce or animate matching video frames.

#### 3.2.3.2 Autoregression on latent codes to measure surprise

Autoregressive models are commonly used on time series and see also some use for other types of ordered sequences such as images, but they can also be used on sequences that are usually ordered such as the features of a latent representation.

Indeed, Abati et al. 2019 use an autoregressive model on the latent representation of an autoencoder. This is motivated by the fact that, as we have said previously, autoregressive models are often probabilistic models, which is the case in this work.

As shown in Figure 3.13, the autoencoder first produces a latent representation, as usual. The autoregressive model, called the estimator, then tries to predict the features one by one. More precisely, the estimator yields a conditional probability distribution over bins of values the features can take. Then, because the estimator yields these conditional probability distributions for each feature, we can estimate a value of likelihood of the latent representation.

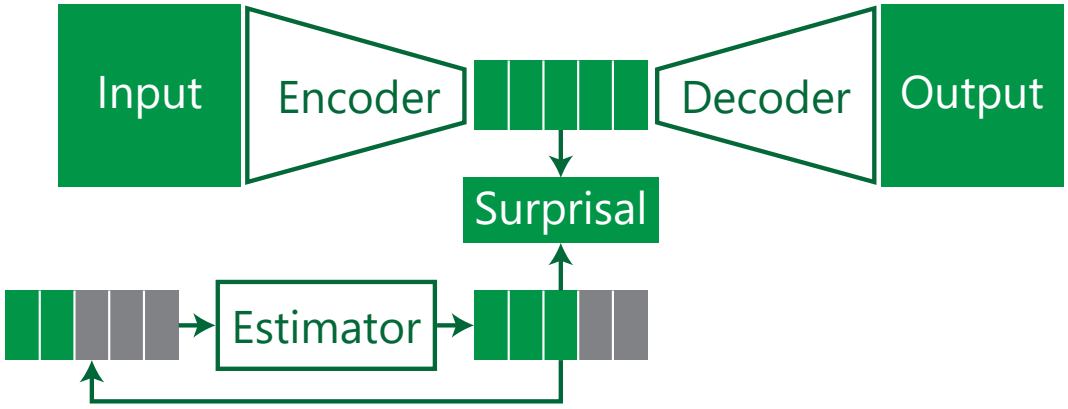


Figure 3.13 – The autoencoder works in the same way as usual. The estimator tries to predict the latent representation yielded by the encoder feature by feature. During training, the encoder and the estimator aim at minimizing the surprisal of these latent representations. During anomaly detecting, this surprisal is used with the reconstruction error of the input to produce an anomaly score.

In their work, Abati et al. 2019 train their model to both minimize the usual reconstruction error and maximize the likelihood of the latent representation. This means that, not only does the estimator learn to estimate the likelihood of a latent representation, but the autoencoder is also trained to produce latent representations the estimator can work

with. In other words, while latent features are usually unordered, the encoder is trained to produce ordered latent features.

During anomaly detection, the reconstruction and likelihood are normalized and then joint to compute the anomaly score. In doing so, they were successful in detecting anomalies in two toy datasets and two surveillance videos, namely the UCSD Pedestrian 2 and the ShanghaiTech Campus datasets.

Overall, for anomaly detection, the concept of using autoregressive models on latent representation is interesting, both for the ability to estimate the likelihood of a latent representation and train a model to treat them as ordered sequences of features.

### 3.2.4 Energy-based models

Energy-based Models (EBMs) (LeCun, Chopra, et al. 2006) are models that take two — or more — inputs and yield a single scalar value, called energy. These models learn an energy function that maps the inputs to an energy value depending on the compatibility between the inputs. The energy value should take low values when the inputs are compatible between each others and should take higher values when they are less compatible.

For example, it is possible to build an energy-based classification model, where the model estimates the compatibility between the sample and the label it is presented with, as illustrated in Figure 3.14 (A). Once the model is trained, it is possible to retrieve the most compatible label for the input using gradient descent to minimize the energy and using only the label as the parameters to fit.

Similarly, we can easily build energy-based GANs but this time the label never changes: the label is always set to "real". This way, the model measures the compatibility between the sample (real or generated) with the label "real", as shown in Figure 3.14 (B).

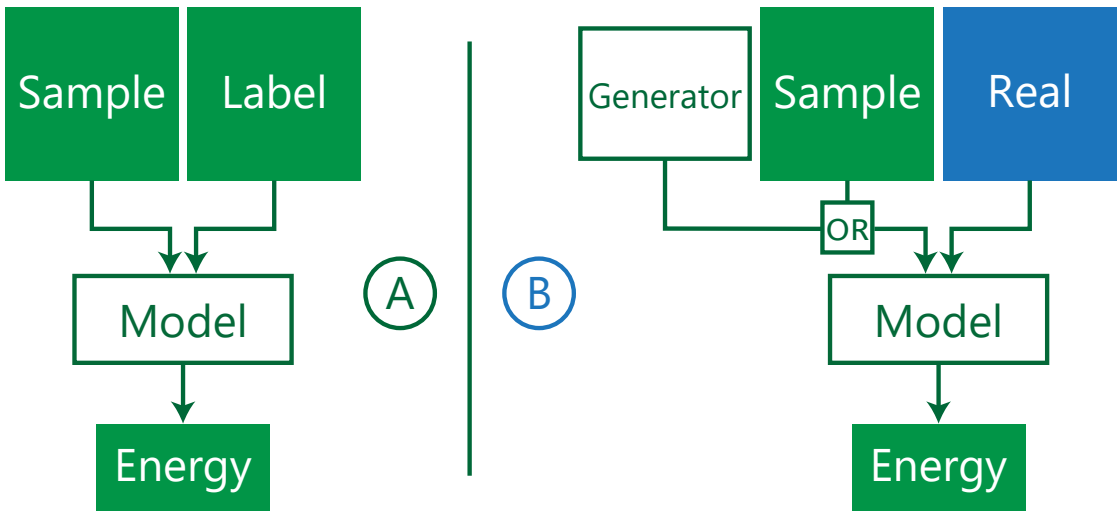


Figure 3.14 – (A) An energy-based classifier measures the compatibility between a given label and a sample. (B) An energy-based GAN measures the compatibility between the constant label "real" and a sample. In both cases (A) and (B), if both inputs are compatible, the model should yield a low energy value. The less compatible the inputs are, the higher the output energy value should be.

Note that we have not described yet the architecture of the energy model. Autoencoders are made of an encoder and a decoder, GANs have a generator and a discriminator and so on, but energy models can take any shape as long as they output an energy value.

For example, the architecture of the energy model could be the one of a discriminator or an encoder, reducing the input space to a single scalar at the end. But it could also be made of an entire autoencoder, as J. Zhao, Mathieu, and LeCun 2016 did. In this work, the energy value is the mean error between the autoencoder's input and output. Thanks to this, EBMs allow for a lot of freedom when designing the energy model.

Note that the principles of EBMs should seem familiar to a certain extent. In anomaly detection model, we aim at giving the higher score possible to anomalies while giving the lowest score possible to normal samples. In a way, we are trying to learn an energy model that yield high energy values for anomalies and low energy values to normal samples. However, the comparison only seems to hold after the model has been trained, as training an anomaly detection model is best performed in a unsupervised or semi-supervised manner as we discussed in Section 3.1.

Nonetheless, we are not only interested in anomaly detection, we are also interested

in —joint— multimodal modeling of human behavior. With this objective, EBMs become interesting to us, as they can be trained to produce joint multimodal latent representations for our data, reducing the dimensionality the data to a more manageable size and increasing the expressivity of the data.

For example, Owens and Efros 2018 developed a model for audiovisual data that is, in essence, an EBM. In their work, they train a model to measure the compatibility between the audio track and the video track. They do so by randomly shifting the audio by a few seconds, forward or backward, with a probability of 50%. The model is then trained to detect whether the audio and video are synchronised or not. This model is illustrated in Figure 3.15. This is an application of self-supervised learning, as described in Section 3.1.4.

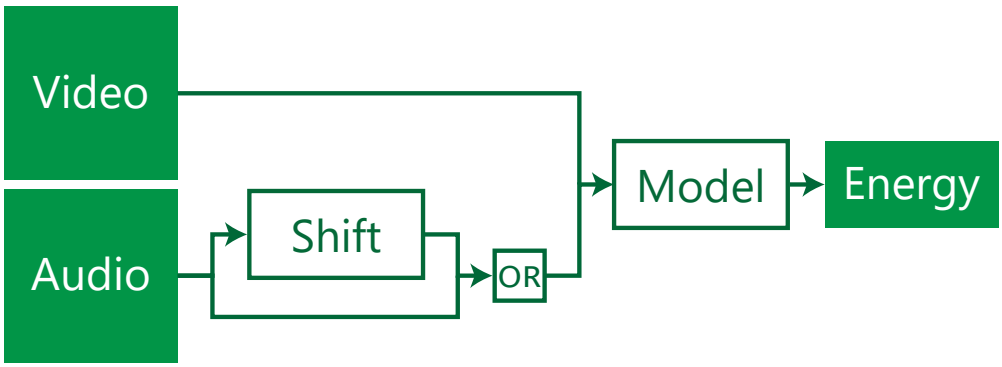


Figure 3.15 – The audio track from a video is shifted with a 50% probability, making the audio track incompatible with the video track about half of the time. The energy model aims at giving low energy values when the tracks are synchronised and higher energy values when they are not.

But Owens and Efros 2018 were not interested in having a model that detects if the audio is synchronised with the video, they were interested in what it could teach the model and the latent representation it would generate to solve the task. Once the model had been trained, they transferred it on three downstream tasks: spatial sound source localization, audiovisual action recognition and on/off screen audio separation. The transferred model was successful on all three downstream tasks.

This particular work motivates us to use EBMs to learn joint multimodal human-behavior dependent representations in order to facilitate the detection of anomalous human behaviors.

### 3.2.5 Meta-learning models

A meta-learning model is a model that learns to learn. Usually, deep learning models tend to be trained once on a single dataset and are therefore specialised for this dataset. Contrarily, meta-learning models can be trained on a set of relevant datasets and, once trained, to quickly specialize on a single dataset.

Meta-learning models are also sometimes referred to as few-shot learners, one-shot learners and zero-shot learners, depending on the situation. By leveraging their learnt ability to learn and their prior knowledge, meta-learning models can learn from only a few to no samples of the target dataset.

In 2014, Graves et al. developed a deep learning architecture inspired by Turing Machines, adding an external and differentiable memory to a neural network. They called instances of this architecture Neural Turing Machines (NTMs) (Graves, Wayne, and Danihelka 2014). As shown in Figure 3.16, NTMs write, read and erase data from this external memory at each step.

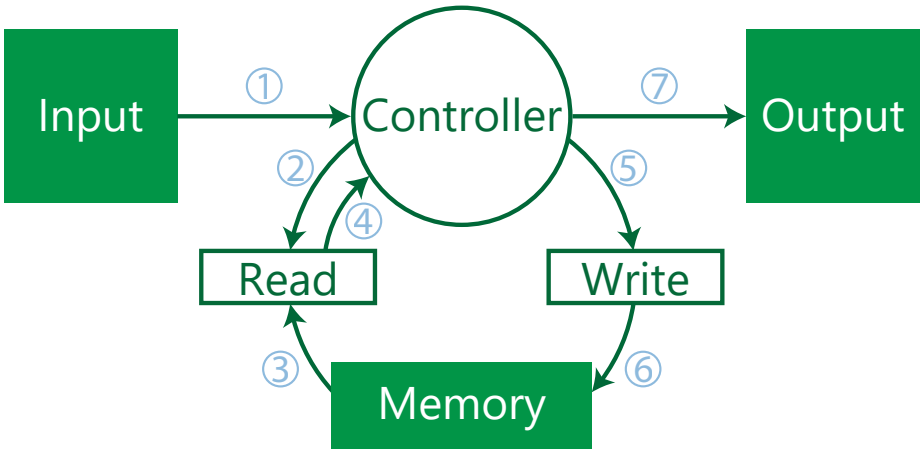


Figure 3.16 – (1) The input is fed to the controller, responsible for addressing the memory. (2) The controller produces read heads. (3) The read heads are used to address the memory. (4) The retrieved data from the memory is fed to the controller. (5) The controller now produces write heads. (6) The write heads are used to write and erase data from the memory. (7) The controller produces the input based on the input and data retrieved from the memory.

However, at this moment, NTMs are only tested on their new found ability to write,

read and erase data from an external memory. It is in 2016 that Santoro et al. 2016 demonstrate their ability to perform meta-learning. Santoro et al. also introduce the notion of Memory-Augmented Neural Networks (MANNs), a class of architectures with augmented memory capacities, such as NTMs.

In their work, Santoro et al. 2016 train a MANN to perform classification on the Omniglot dataset (Lake, Salakhutdinov, and Tenenbaum 2015), a dataset containing 1623 different handwritten characters from 50 different alphabets. However, there is a twist in how the classification task is formulated. The model is presented with a sequence of tuples $(x_t, y_{t-1})$ where $x_t$ is the $t$-th sample and $y_t$ is the label of the $t$-th sample. At each step of this sequence, the model tries to predict the label of the current sample. The important change happens once the sequence is complete: the labels are randomly shuffled. For example, 0s receive the label 7, *etc.* meaning the MANN has to re-learn every time a new label for each classes. In this experiment, when MANNs received samples from 5 classes, they were able to reach an accuracy of 83% on just the second presentation of a sample from a class.

MANNs later saw new iterations, such as Differentiable Neural Computers (DNCs) (Graves, Wayne, Reynolds, et al. 2016), improving on the way memory is accessed, and the Kanerva Machine (Y. Wu, Wayne, Graves, et al. 2018; Y. Wu, Wayne, Gregor, et al. 2018), a memory-augmented generative model inspired by Kanerva's sparse distributed memory.

In the framework of anomaly detection, meta-learning models interest us for different reasons: Their ability to learn from only a few samples can help adapt quickly to a change in the context, or deploy to a completely new context with minimal costs. They also open up new horizons for training other models, with the possibility of better teaching to models or selecting hyper-parameters.

#### 3.2.5.1 The Kanerva Machine and attractors

Developed by Y. Wu, Wayne, Graves, et al. 2018, the Kanerva Machine is a memory-augmented generative model set in a probabilistic framework. This allows the Kanerva Machine to perform one-shot generation, or few-shot generation, where the model generates new samples after seeing only a few samples drawn from a few classes.

One of the important aspects of the Kanerva Machine is its iterative reading mechanism. The output of the model is repeatedly fed back as the input for a few steps. Experiments conducted by Y. Wu, Wayne, Graves, et al. 2018 indicated this iterative mechanism

improves the output over the iterations, as the latent representation seems to converge towards a representation stored in memory. This reliably improved both denoising and sampling.

In their second work on the Kanerva Machine, Y. Wu, Wayne, Gregor, et al. 2018 aim at improving the robustness of this retrieval procedure of a pattern stored in memory, especially in the presence of interference due to the presence of other patterns stored in memory or due to the presence of noise. They consider the usage of attractors, a theoretically well-founded method for this task. Note that, in the case of the Kanerva Machine, we are only interested in simple attractors — *i.e.* fixed points — and not in more complex attractors, such as limit cycles and strange attractors.

Assuming the model is a function $f$ and $x$ the input, $x$ is a fixed point if $x$ stays the same after one iteration: $f(x) = x$ (in other words, $x$ is a fixed point for $f$). Such a point can be used as an attractor and corresponds to a stored pattern in memory. As shown in Figure 3.17, these attractors can be used to simulate attractor dynamics, making nearby points iteratively converge to these attractors.

In practice, to converge to a fixed point, an energy function $\mathcal{E}(x, w)$ described in Equation 3.1 is minimized iteratively. This is done by optimizing the input $x$ and the addressing vector $w$ alternately, where $w$ is a vector used to address the external memory $M$. This energy function is made of two terms. The first term aims at maximizing the probability of the input $x$ given the addressing vector $w$ and the external memory $M$. The second term is basically the same as with VAEs. Indeed, $w$ is not obtained directly by encoding $x$ but is rather sampled from $q_t(w)$ — which is obtained from $x$ — where $t$ is the index of the iteration (of the convergence iterative process to a fixed point).

$$\mathcal{E}(x, w) = -\langle \ln p(x|w, M) \rangle_{q(M)} + D_{KL}(q_t(w)||p(w)) \tag{3.1}$$

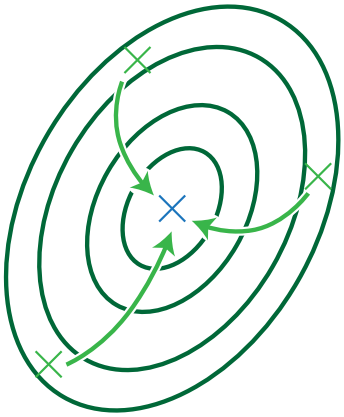

Figure 3.17 – × is a fixed data point and is used as an attractor. By simulating attractor dynamics, the nearby × can be iteratively attracted to ×.

While, to our knowledge, this has never been tested before, we believe these dynamics to be relevant for anomaly detection. We would expect such a model to learn to create clusters around these attractors. Not having been observed during training — not meta-training —, anomalies should not have their matching patterns stored in memory and we would expect these patterns to fall away from the clusters, in a similar fashion as with conventional clustering methods.

Moreover, this method would provide us with a potentially better metric than simply computing the distance in the latent space and give us the possibility to take into account the local curvature of the learnt latent space. One could use the learnt ability of the model to simulate attractor dynamics and compute the number of iterations required to make a new observation converge towards a stored pattern in memory.

Using Equation 3.1, the algorithm can iteratively converge towards a fixed point (*i.e.* a stored pattern). Furthermore two thresholds are defined, one for the energy function and one serving as a limit to the number of iterations. If one of these two thresholds is exceeded, the iterative convergence process is stopped. Then, an anomaly score can be determined either from the number of iterations taken or the value(s) of the energy function, or both.

## 3.3 Evaluation of an Anomaly Detection System

As we have highlighted in Section 2.3.2.4, there exist several metrics to evaluate an anomaly detection system's performance. These metrics co-exist because none of them is really better than the other ones, as they measure different aspects of an anomaly detection system's performance and are more or less relevant depending on the application case.

In some application cases, the user may prefer a very sensitive anomaly detection system. This system would raise alarms as soon as there is the slightest suspicion of an anomaly. On the other hand, the user may not have the appropriate resources to handle all suspected anomalies and would be more concerned to have a system with a low false positive rate.

In this section, we will discuss the main evaluation metrics found in the literature and other metrics that are usually derived from them. But before we begin, we must come back on something we have talked about in the previous sections and that is common to all those metrics: the notion of anomaly score and their thresholds.

#### 3.3.0.1 Anomaly scores

Anomaly detectors typically do not output a single Boolean value to answer the question: "Is this sample an anomaly ?" Instead, they output a scalar, typically normalized between 0 and 1, where 0 indicates the sample is unlikely to be anomalous and 1 indicates it is very likely to be anomalous. Note that these values are likelihoods, not necessarily probabilities.

Also note that, contrarily to us, in the literature authors may occasionally refer to the "regularity score" or "normality score", which is the opposite of the anomaly score. However, the same basic principles apply.

#### 3.3.0.2 Anomaly score thresholds

Nonetheless, the anomaly score being a scalar, it does not yet answer our initial question: "Is this sample an anomaly ?" This is where the notion of thresholds becomes important. If the anomaly score is above a given threshold, then the answer to our question will be yes, and no otherwise.

The question then becomes "How is this threshold defined ?" This question is not ours to answer, but a decision that will be taken by the final user. On our side, we evaluate the system on a range of thresholds, typically a hundred thresholds uniformly spread between

0 and 1. This helps us better estimate a system's performance. This also helps inform the final user about how their system will behave depending on the threshold they select and the errors made by the system.

All the following evaluation metrics use this notion of thresholding. Because of this, they all yield a curve that are direct or indirect functions of said thresholds.

#### 3.3.0.3 Area Under the Curve

All these evaluation metrics can be summed up individually by a single scalar value. Since all evaluation metrics produce curves, the most commonly considered scalar value is the Area Under the Curve (AUC).

#### 3.3.0.4 General performance metrics

Finally, we note that there exist metrics relevant to anomaly detection that are not directly related to it. For example, the computational cost, the execution time, the scalability, the portability are all important elements to consider when building an anomaly detection system: A video surveillance system is useless if it takes one day to analyse a single video frame, even if it beats other systems in all other areas. However, these general metrics also depends on other elements the anomaly detection method is not responsible for, such as the hardware. Because of this, we will only be making sure our methods can run in real-time on hardware affordable for regular consumers.

### 3.3.1 Sensitivity and Specificity

Sensitivity, also known as the True Positive Rate (TPR), measures the fraction of all known anomalies that were identified as such. Its counterpart is known as the False Positive Rate (FPR), the fraction of false alarms.

Specificity, also known as the True Negative Rate (TNR), measures the fraction of all normal samples that were identified as such. Its counterpart is known as the False Negative Rate (FNR), the fraction of missed anomalies.

We draw typical instances of sensitivity and specificity curves as functions of the anomaly score threshold in Figure 3.18.

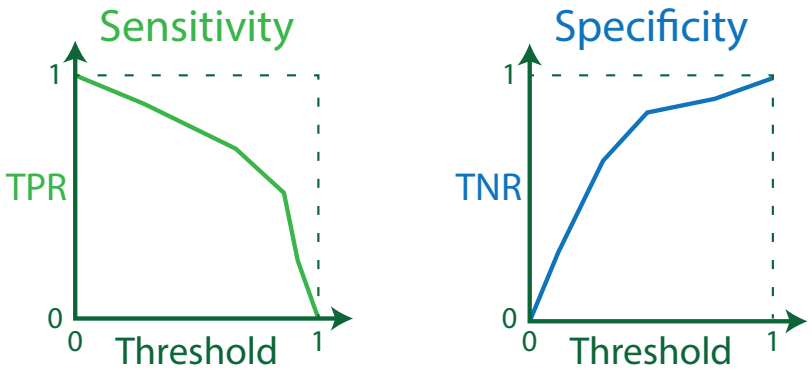


Figure 3.18 – As the threshold of a system increases, its specificity increases but its sensitivity decreases. An overall higher sensitivity and specificity indicates an overall better system.

A system with a high sensitivity will rarely miss any anomaly. However, depending on its quality, it runs the risk of classifying a high amount of normal samples as anomalous. On the other hand, a system with a high specificity will rarely misclassify any normal samples but — depending on its quality — runs the risk of missing anomalies. When building an anomaly detection system, the goal is for it to reach a sensitivity and a specificity as high as possible at the same time. However, there is a trade-off to be made between sensitivity and specificity.

At a given anomaly score threshold, a system A surpasses a system B in anomaly detection if both the sensitivity and specificity of A are above B's. If for all thresholds, A is always above B both in terms of sensitivity and specificity, then A always surpasses B in anomaly detection.

### 3.3.2 Receiver Operating Characteristic

Given two curves, the TPR and the FPR, as functions of the same anomaly score thresholds, the Receiver Operating Characteristic (ROC) curve plots the TPR as a function of the FPR, as illustrated in Figure 3.19.

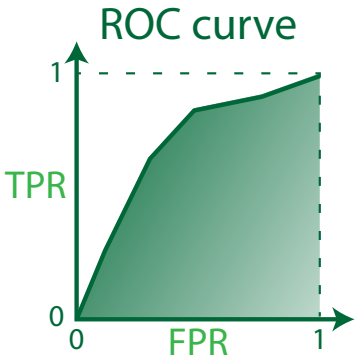


Figure 3.19 – The ROC curve is obtained by plotting the TPR as a function of the FPR using the same detection thresholds. The AUC presented in a gradient of green is a common metric to measure a system's performances. The higher the AUC of the ROC curve, the better.

Overall, the area under the ROC curve is by far the most reported metric in the literature. Due to the length constraints usually imposed by publishers on articles, the ROC curves themselves are more rarely reported.

#### 3.3.2.1 ROC Convex Hull (ROCCH)

Provost and Fawcett 2001 introduced a simple and elegant improvement over the ROC curve: using its convex hull. As shown in Figure 3.19, the convex hull is the convex shape the closest to the curve. To build the convex hull, any point that is below a line formed by any pair of other points is removed.

In practice, given an anomaly score, this is done by randomly picking one of the two closest points on the convex hull, depending on the distance to these two points. The closer the score is to one of the points, the more likely this point will be picked over the other.

Because the ROC cannot start below the point (0, 0) and end below the point (1, 1), the worst convex hull a model could get is a straight line joining these two points. Such a model would have an area under the ROC curve of 0.5, which would be the equivalent of using a random anomaly detector.

For example, let's take a bad system which would only consider samples to be anomalous if their anomaly score is equal to 1. This system would have a FPR and a TPR of 0,

until the threshold of 1 is reached, where these two values would become 1. Without the convex hull, the AUC is 0. With the convex hull, we draw a line between the point (0, 0) and the point (1, 1). If we receive an anomaly score of $x$, where $x \sim \mathcal{U}(0, 1)$, then we have probability of picking the point (1, 1) that is equal to $x$. Since $\mathbb{E}(x) = 0.5$, we fall back onto a random anomaly detector, classifying samples as anomalies with a probability of 0.5, resulting in TPRs and FPRs with an average of 0.5, and an area under the ROC curve of 0.5 as well.

More interestingly, Provost and Fawcett 2001 demonstrated that the ROCCH can be used to take advantage of having several systems with different ROC curves. This allows to use each system on the thresholds it performs the best. We illustrate this use in Figure 3.20.

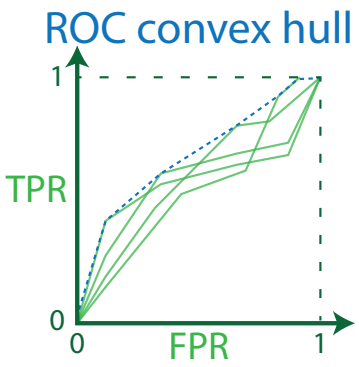


Figure 3.20 – The convex hull is illustrated in blue dashes. It is computed from all sub-systems ROC curves and yields a better overall system by indicating which sub-system should be used depending on the anomaly score.

#### 3.3.2.2 Equal Error Rate (EER)

As others curves, the ROC can be summarized by the AUC but it can also be summarized by an other single scalar value called the EER. The EER is the value where the FPR and the FNR become equal. Graphically, this point is easily found on a ROC curve by drawing a straight line from the point (0, 1) and the point (1, 0), as shown in Figure 3.21.

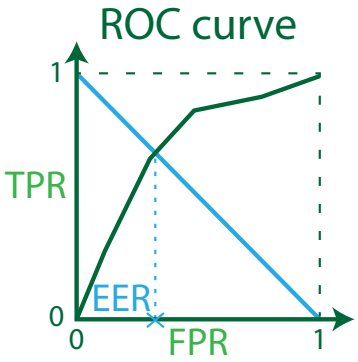


Figure 3.21 – The blue line illustrate all positions where the error rates become equal. Since the ROCCH curve is convex, it can only intersect once with the EER line. The intersection point yields the EER for the system. The lower the EER, the better the system.

Contrarily to metrics such as the area under the ROC curve, a lower EER indicates a better performance and while the area under a ROC curve is situated in the interval $[0.5, 1]$, the EER is situated in the interval $[0, 0.5]$. An EER value of 0 would indicate a system able to perfectly discriminate anomalies from normal samples, having a TPR of 1 and a FPR of 0 for any threshold.

### 3.3.3 Precision and Recall

The precision is defined as the fraction of positives that are true positives: $\frac{\text{TPR}}{\text{TPR}+\text{FPR}}$. This metric informs us about something we were missing until now. It answers the question: "Given that the model flagged a sample as an anomaly, what is the probability this sample really is an anomaly ?"

Let us illustrate the importance of the precision metric with a made-up situation. Let us consider an anomaly detector with a 99.9% accuracy, meaning that it will give the right answer 99.9% of the time. This anomaly detector seems like a really good detector at first sight. What if a sample is flagged as anomalous by our detector, what is the probability this sample is actually anomalous ? The answer is we cannot conclude now, because we do not know the prior probability this sample has to be anomalous.

If we are in a situation where only 1 sample out of $10^6$ is actually anomalous, this means for each $10^6$ samples the system sees, an average of 0.999 anomalies will be correctly

identified but this also means that an average of 999 normal samples will be flagged as anomalous. Overall, when this system will raise an alarm, it will be 1000 times more likely to be a false alarm than a true positive.

While this ratio may be acceptable in some sensitive situations, it will not always be acceptable. For final users that are more concerned about making sure flagged samples are actual anomalies, the system's precision is a critical metric to look at.

#### 3.3.3.1 Precision-Recall (PR)

Before defining the PR curve, we note that the Recall is another name for the sensitivity (or the TPR).

As shown in Figure 3.22, the PR curve is obtained by plotting the precision as a function of the recall, in the same way the ROC curve plots the TPR as a function of the FPR.

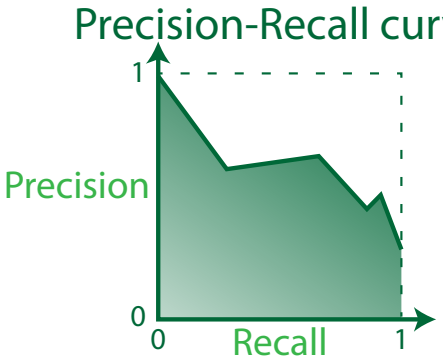


Figure 3.22 – The PR curve is obtained by plotting the precision as a function of the recall using the same detection thresholds. The AUC presented in a gradient of green is sometimes used to measure a system's performances. The higher the AUC of the PR curve, the better.

Overall, the PR curve is more rarely seen in the literature than the ROC curve. Nonetheless, thanks to the precision metric being included, the PR curve gives a more realistic picture of the system's performances when anomalies are significantly rarer than normal samples. This is more likely to be the case in real-world application than in the literature, as commonly used datasets usually have balanced amounts of anomalies and normal samples.

As with the ROC curve, it is possible to summarize the PR curve using the AUC. The resulting scalar value is sometimes referred to as the Mean Average Precision (mAP).

#### 3.3.3.2 Precision-Recall-Gain

However, as demonstrated by Flach and Kull 2015, the area under the PR curve *is fraught with difficulties, mainly because of incoherent scale assumptions*. To correct this problem, they introduce a new metric: the Precision-Recall-Gain. They do this by *plotting PR curves in a different coordinate system, and demonstrate that the new Precision-Recall-Gain curves inherit all key advantages of ROC curves*.

The mapping to a different coordinate system is rather straightforward. Flach et Kull observe the precision value never goes below a certain value they note as $\pi$. Given this observation, they only consider values in the interval $[\pi, 1]$ for both the precision and the recall. To do so, they remap this interval to a normalized interval $[0, 1]$ using an harmonic scale:

$$f(x) = \frac{x_{max} \cdot (x - x_{min})}{(x_{max} - x_{min}) \cdot x} \tag{3.2}$$

And since we know $x_{min} = \pi$ and $x_{max} = 1$, this equation can be simplified:

$$f(x) = \frac{x - \pi}{(1 - \pi) \cdot x} \tag{3.3}$$

We illustrate the resulting remapping in Figure 3.23.

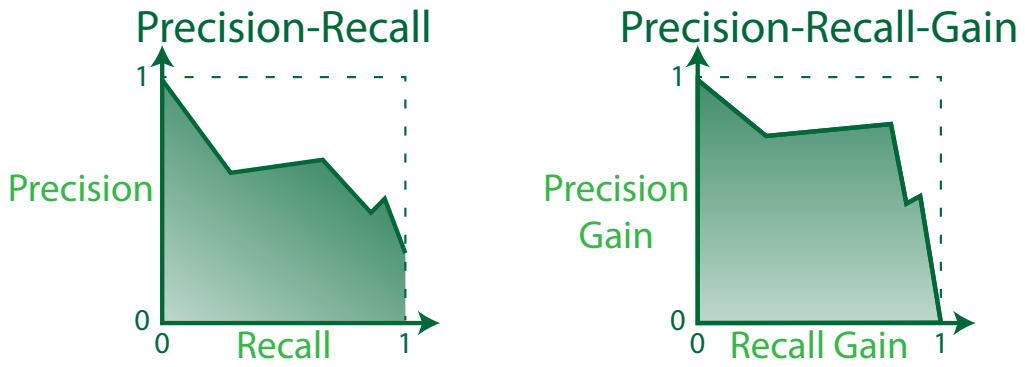


Figure 3.23 – Contrarily to the precision, the precision-gain always goes down to 0 when the recall-gain reaches 1. Because of the harmonic scale, points closer to (0, 0) are dragged further away from their previous position.

They underline two benefits to this new method. First, this allows the use of convex hulls for the Precision-Recall-Gain curve, as we saw with ROC curves. Second, they demonstrate that the area under Precision-Recall-Gain curves leads to better model selection thanks to less biased scores.

Chapter 4

# Main contributions

We now address the main outcomes of this thesis in the form of three anomaly detection methods: the Interpolating Autoencoder (IAE), the Variational Interpolating Autoencoder (VIAE) and the Clear and Concise (CnC) model. Before detailing these methods, we describe the common building blocks shared by all the methods we have developed in the scope of this thesis. Then, we describe each of the three aforementioned methods using the following pattern:

First, we start with the reasons that motivated us to develop each proposed method. This is where we frame the problem we are trying to solve and this gives us the opportunity to describe the thought process that led us to this solution.

Second, we describe the underlying architecture of the resulting models. This is mostly helpful to better understand the next part.

Third, we present the objective function of the model. This function yields a single scalar value the model has to minimize during training. For all presented cases, this objective function is the heart of the method, as it conditions the way normality is being modeled.

Fourth, we detail how anomalies can be detected once the model has been trained. For all methods, the architecture and objective function give us several options for anomaly detection. We present the different reasons one might have for using these options.

Fifth, for the IAE and the VIAE, we detail the experimental protocols we used in order to evaluate the methods. These experimental protocols are as similar as possible to the ones found in the literature in order to ensure a fair comparison of the different methods.

Sixth, we present the results of the method. Since we identified several valid options, we evaluate and compare some of them. With the lessons drawn from Section 2.3, we aim at making these evaluations as thorough as possible.

Seventh, we come back to the method itself to discuss its strengths, its flaws and what would we do next to improve it, given the opportunity.

Finally, we detail how to adapt these methods for multimodal anomaly detection. Indeed, while none of these methods were evaluated on multimodal anomaly detection, all of them were designed with joint multimodal modeling of human behaviors in mind. Unfortunately, since there is not much literature on multimodal anomalous human behavior detection to compare to, we built from and compared our methods to monomodal anomalous human behavior detection methods.

## 4.1 Building Blocks

As shown in Figure 4.1, deep learning models can be viewed as graphs, where groups of operations are represented by nodes and data are by edges.

Most of these nodes are called "layers" and can be made of a wide variety of operations, including other layers. These layers and operations are the building blocks of our models. While it is not required to fully understand these layers to understand our work, the insights we will give will help understand some of the challenges we faced, notably when modeling multiple modalities.

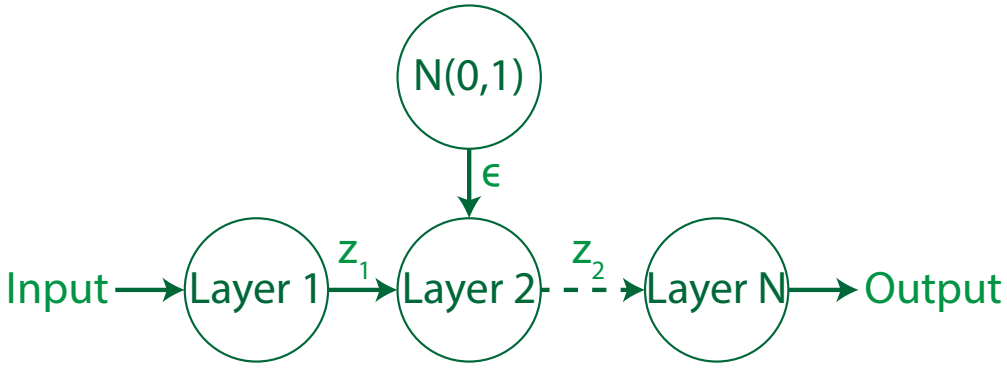


Figure 4.1 – Representation of a model with $N$ layers as a graph. Nodes represents operations while edges represent data. The sampling of the normal distribution $\mathcal{N}(0, 1)$ displays an operation that is not a learnt layer and shows that a layer can have multiple inputs (and multiple outputs).

### 4.1.1 Rank of a tensor

Tensors are multi-dimensional arrays. The rank of a tensor indicates the number of dimensions it has. For example, a matrix is a tensor with a rank of 2. Here are three concrete examples: a grayscale image has a rank of 2 — height and width —, an image with color has a rank of 3 — height, width and channels — while a waveform audio has a rank of 1 — time — if it has only channel and a rank of 2 if it has more than one channel.

### 4.1.2 Convolutional Neural Networks

Convolution filters have been historically used to accomplish image processing tasks such as blurring, edge detection, keypoint detection, *etc.* These filters are also referred to as kernels and are tensors of any rank greater or equal to 1.

Common examples for kernels for image processing include the Sobel, Laplacian and Gaussian kernels. The Sobel and Laplacian kernels are mostly used for edge detection while Gaussian kernels are used for blurring. As shown in the 1-D example presented in Figure 4.2, these kernels can be seen as sliding windows. The window is centered on an element — a pixel in the case of images — and computes a weighted sum of the neighbors of this element, the weights depending on the selected kernel.

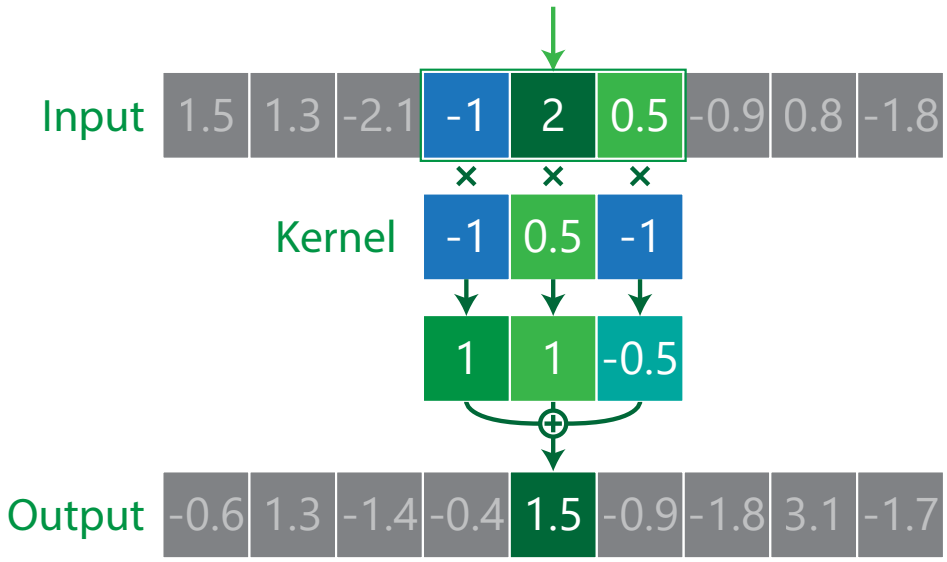


Figure 4.2 – A 1-D kernel being applied to compute an element of the output sequence shown at the bottom. In this case, the kernel uses a window of 3 elements in the input sequence. The output is a weighted sum of those 3 input elements, the weights $(-1, 0.5, -1)$ being defined by the kernel.

However, once they have been chosen, the values of these filters are fixed. Instead of fixing these weights, Convolutional Neural Networks (CNNs) learns them during training. While CNNs were originally designed for images and hand written character recognition (LeCun, Boser, et al. 1989), they can be used for any sequential data, such as text, audio, images and videos.

CNNs can be used on data regardless of the rank of the input, as long as this rank is at least 1. However, the rank of the kernel changes with the rank of the input. This implies that a kernel meant for images (a 2-D kernel) cannot be used as-is on videos and audio. Videos would require a 3-D kernel and waveform audio would require a 1-D kernel.

Additionally, CNNs usually have many kernels — one per input feature and per output feature — and a CNN trained on audio would probably not be suitable on video anyway, as their features are very different. These limitations will therefore impact the way we approach the joint modeling of multiple modalities as we cannot simply use the same CNN architecture for two different modalities.

### 4.1.3 Residual Blocks

Introduced by He et al. 2016 in 2015, residual blocks introduce a skip connection after one or more CNNs. This means, if we consider $x$ to be the input of the layer $f$ and $f(x)$ the output of this layer, the output of the residual block $f_{res}(x)$ is given by:

$$f_{res}(x) = x + f(x) \tag{4.1}$$

By adding this skip connection, residual blocks aim at easing the vanishing/exploding gradient problem. This is a problem that arises when a neural network is too deep, as the norm of the gradient basically exponentially increases or decreases with the number of layer: $a^{n_{layers}}$. If $a < 1$, then the gradient will vanish and the neural network will not receive any update. If $a > 1$, the gradient will explode and the neural network weights will explode as well, making the model unstable. The introduction of skip connection gives a straight way from the input to the output where $a = 1$.

As we illustrate in Figure 4.3, there exist multiple variants of residual blocks. We display the two variants we used in this thesis: the FixUp (H. Zhang, Dauphin, and T. Ma 2019) and the SkipInit (De and Smith 2020) variants. In short, the FixUp aim at improving the control over the norm of the gradient and SkipInit goes even further in this direction. We used FixUp in the early methods of this thesis, such as the Interpolating Autoencoder (IAE) and we then changed to the SkipInit when it was introduced later.

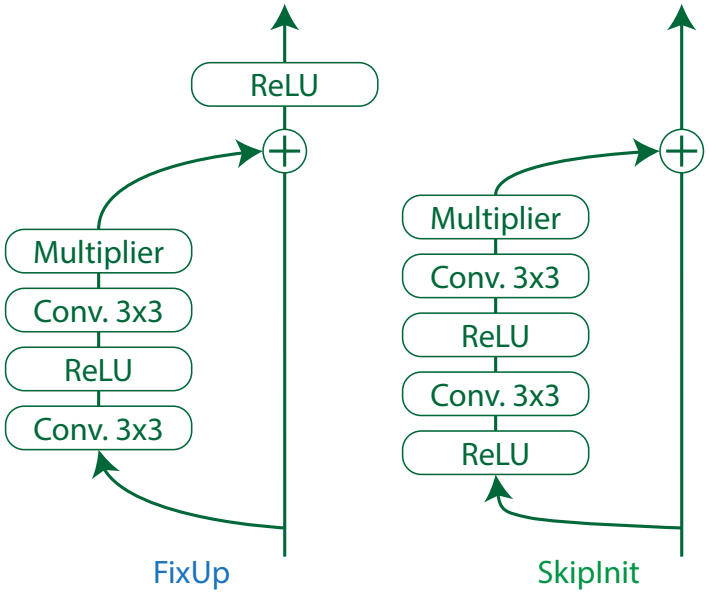


Figure 4.3 – Representations as graphs of single basic blocks of the FixUp (H. Zhang, Dauphin, and T. Ma 2019) and the SkipInit (De and Smith 2020) variants of residual blocks (He et al. 2016). The ReLU activation function (Agarap 2018) is a simple non-linear function clipping values below 0.

### 4.1.4 Flatten and Reshape

As we mentioned before, we cannot directly use different modalities together as is because of their different ranks and different number of features. Ideally, we would like to be able to sum them up or at least concatenate them into a single tensor. This single vector would then be fed to a layer in order to model these modalities in a joint manner.

This is where the flatten operation and its opposite operation the reshape — or rather un-flatten — operation come in.

The process of flattening the dimensions of a tensor is the act of considering these multiple dimensions as only one, as illustrated in Figure 4.4. For example, flattening all dimensions of a 4-D tensor such as a video with shape $[time, height, width, channels]$ turns it into a 1-D tensor with shape $[time \cdot height \cdot width \cdot channels]$.

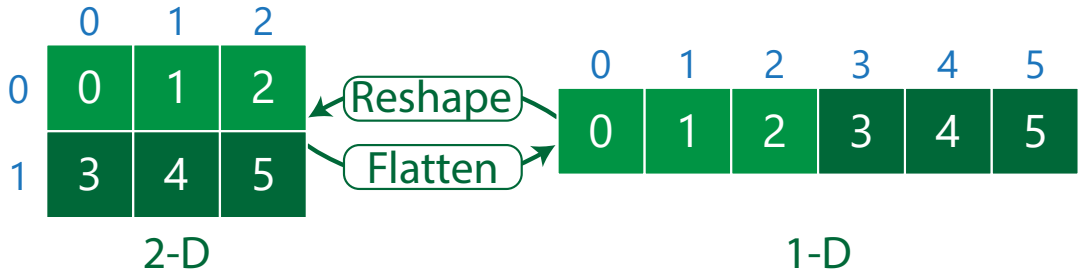


Figure 4.4 – A 2-D tensor being flattened into a 1-D tensor, and vice-versa.

It is worth noting that we do not have to flatten all dimensions every time. Using the previous example, we can decide to flatten all dimensions except time, yielding a 2-D tensor with shape $[time, height \cdot width \cdot channels_{video}]$. Assuming an audio track represented by a tensor with shape $[time \cdot n, channels_{audio}]$, we can reshape to $[time, n \cdot channels_{audio}]$ and concatenate it with the tensor representing the video track, yielding a 2-D tensor with shape $[time, height \cdot width \cdot channels_{video} + n \cdot channels_{audio}]$.

Note that the inverse process is possible: split the single tensor into two, one for each modality, and reshape both tensors to their respective original shape. If the flattening was mostly useful for the encoder in an autoencoder, this will be particularly helpful in the decoder.

## 4.2 Interpolating Autoencoder

The IAE is a method developed for anomaly detection in time series and has been evaluated on surveillance video datasets. This method has been the subject of a publication in the SMC IEEE 2020 conference (Durand de Gevigney et al. 2020).

### 4.2.1 Motivation

As we have explained in Section 3.2.3, autoregressive models — and transformers more specifically — can be interesting for modeling human behavior due to their ability to predict the next element of a sequence.

Additionally, as we have seen in Section 1.2.1, one of the way anomalies can be defined is by their relative unpredictability compared to normal samples.

We started the design of the IAE based on these observations and formulated the following hypothesis: A well-trained transformer will only be able to predict future human behaviors based on present — and past — observed behaviors if all observed behaviors are normal. Therefore, a failure to predict future behaviors would indicate an anomaly.

However, in our early experiments, the high dimensionality of surveillance video prevented us to produce a well-trained model. One striking observation was that the transformer would basically copy the last frames instead of trying to predict future frames. We believe this is due to the fact that such a solution was locally optimal for the objective we gave to the model — *i.e.* minimizing the reconstruction error — and that this local optimum was likely to be found early. The model appeared to always go in this direction.

We then tried to add a penalty to the model when it simply copied the input, both in the input space and the latent space, in separate experiments. This only seemed to make it worse: If the penalty was not big enough, the model would stay the same. If the penalty was big enough, the model would just fail to produce anything coherent and degenerated.

However, there was a significant difference between our method and transformers in NLP: the encoders and decoders used to project the data from the input space to the latent space. Traditionally, transformers use a lookup table — as the one shown in Figure 4.5 — to project the label of a token to its corresponding embeddings. This is a relatively simple operation that does not add a lot of complexity to the model.


Figure 4.5 – A simple lookup table with only 3 tokens: "cat", "dog" and "hello". The lookup table returns a vector depending on the selected token.

In our case, we had a full additional autoencoder trained jointly with the transformer as shown in Figure 4.6. First, the encoder would project the video from the input space to the latent space. Then, the transformer would try to predict the future in the latent space. Finally, the decoder would decode the latent representation predicted by the transformer, making it back to the input space.


Figure 4.6 – A transformer working on latent codes produced by an encoder. The transformer outputs updated latent codes that are then decoded back into the input space. The addition of an encoder and decoder should unburden the transformer during training.

The IAE method was then created to fulfill the following objective: pre-train an autoencoder in a temporally constrained manner, in order to facilitate the job of the transformer and reduce its chances of following the same local minimum as before.

IAEs will be trained to produce linearly interpolable latent codes. Given a point in the past and a point in the future, a well-trained IAE should be able to draw between these two points a line we can sample from, as shown in Figure 4.7.

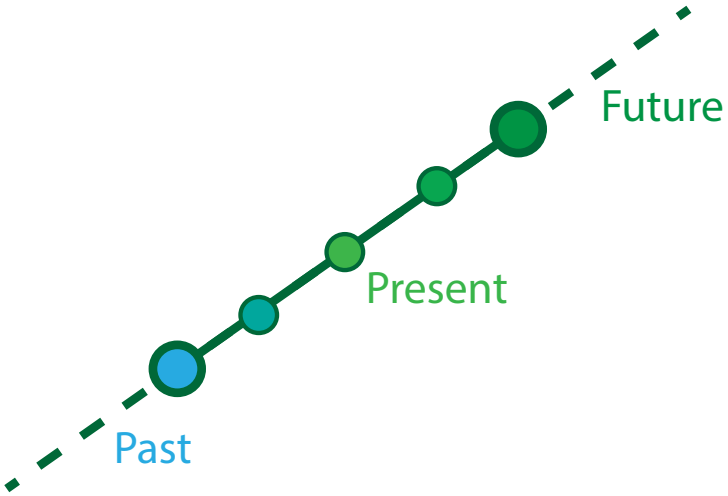


Figure 4.7 – Given two points, one in the past and one in the future, a well trained IAE should be able to sample along the line formed by these two points.

However, if we then take a point in the past and a point in the present,should we not be able to draw a line as well and predict a likely future using this line ? Moreover, do we really need to train a transformer to do something as simple as a linear interpolation

between two points ? This is question that made us want to evaluate the IAE on its own for anomaly detection in surveillance video.

### 4.2.2 Architecture

The base architecture of an IAE has been defined from the following observation: multiple modalities — such as audio and video — can be temporally aligned and a joint latent representation should represent the same duration in both modalities.

This mainly implies it is preferable to consider small segments of our modalities instead of the individual elements that compose these segments. For example, it is more helpful to consider one second of a 25fps video as a single element, instead of 25 individual images, because we can align this second of video with its corresponding second of audio, as shown in Figure 4.8. In this case, we can obtain a tuple of aligned segments $(video_n, audio_n)$ where the full input $(video, audio)$ is made of $N$ of these tuples.

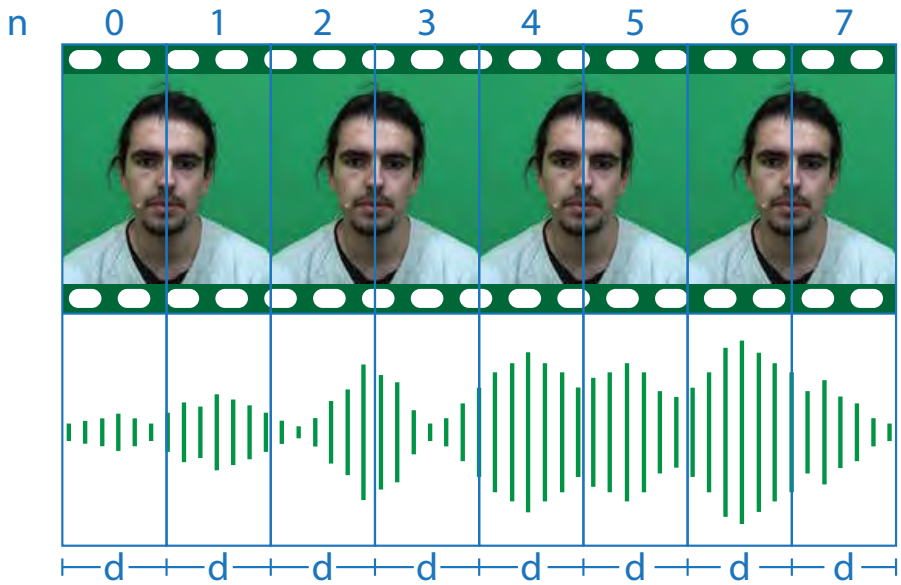


Figure 4.8 – The video and audio tracks are split into $N = 8$ aligned segments of the same duration $d$. IAEs consider each resulting segments as single elements of a sequence made of $N$ elements. Images come from the Emo&ly dataset.

Additionally, a single frame carries little — *e.g.* video — to no temporal information — *e.g.* audio waveform —. For these reasons, IAEs always encode a few frames at once.

### 4.2.3 Objective function and training

IAEs are basically trained to predict the present based on the past and the future. For simplicity sake, we will consider the IAE is only used on video for the moment.

Given a video $x$ with $L$ frames, we slice this video into $N$ segments $x_n$ of equal length $T$, so the total video length is $L = N \times T$. Given an encoder $e$ and a decoder $d$, let the latent code $e_n(x)$ of the $n^{th}$ video segment be :

$$e_n(x) = e(x_n) \text{ where n} \in [0, N-1] \tag{4.2}$$

Let $f_n(x)$ be the latent code of the $n^{th}$ video segment obtained by interpolating between the first and last segments $e_0(x)$ and $e_{N-1}(x)$:

$$f_n(x) = e_{N-1}(x) \times \frac{n}{N-1} + e_0(x) \times (1 - \frac{n}{N-1}) \tag{4.3}$$

And let the reconstruction using the decoder $d$ of this interpolated latent code be:

$$\tilde{x_n} = d(f_n(x)) \tag{4.4}$$

Then finally, the objective function of the IAE can be defined as:

$$\mathcal{L}_{IAE} = \frac{1}{N} \sum_{n=0}^{N-1} \|\tilde{x_n} - x_n\|^2 \tag{4.5}$$

This means IAE are trained to minimize the reconstruction error between the original video segments $x_{0,1,\ldots,N-1}$ and their reconstruction after interpolation $\tilde{x}_{0,1,\ldots,N-1}$.

The overall interpolation process is illustrated in Figure 4.9.

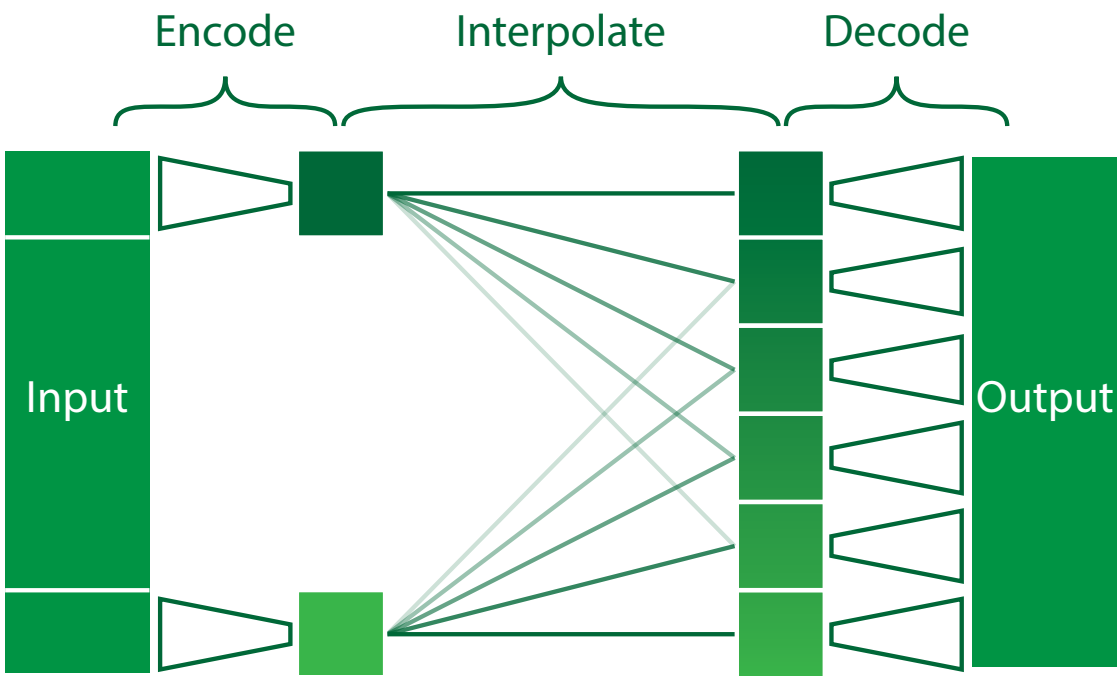


Figure 4.9 – The autoencoding process of an IAE has three steps. 1) Using the same encoder, the past and the future are encoded. 2) To get the intermediate latent codes, linear interpolation is performed between the latent codes of the past and the future. The interpolation factors range from 0 — for the past — to 1 — for the future —. 3) All latent codes are decoded, using the same decoder, and are concatenated along the time axis.

### 4.2.4 Anomaly detection

Assuming we have trained an IAE on a dataset, we can now start detecting anomalies. Since an IAE is an autoencoder, it can be used in the way autoencoders usually are, as we previously described in Section 3.2.1.

However, we can choose to keep or discard the interpolation step at this stage. If the latent code interpolation process is discarded, then the remaining autoencoder can be used either on smaller segments of length $T$ or on longer videos with $L = N \times T$ frames.

In our experiments, we choose to evaluate the IAE both with and without the interpolation process. In order to evaluate these two configurations with the same setup, they are both evaluated on videos with $L$ frames.

For the distance measure between the input and the reconstruction, we choose to evaluate the usage of 4 metrics: the MAE, the MSE, the Peak Signal to Noise Ratio

(PSNR) and the Structural Similarity Index Measure (SSIM). It is worth noting that we felt using the MAE and MSE would be enough to compare the situations with and without the interpolation process, so we did not use the PSNR and the SSIM while keeping the interpolation process.

Additionally, since the learning process of an IAE heavily impacts the learnt latent space, we decided to include a metric to compare samples in the latent space. As illustrated in Figure 4.10, the idea is that if normal samples are mapped to a straight line in the latent space, anomalous samples probably will not. To test this, we compute the Cosine Similarity (CS) between the interpolated latent codes $f_n(x)$ and default latent codes $e_n(x)$:

$$\text{latent-cosine-similarity}(x) = \frac{1}{N-2}\sum_{n=1}^{N-2}\frac{e_n(x)\cdot f_n(x)}{\|e_n(x)\|\|f_n(x)\|} \tag{4.6}$$

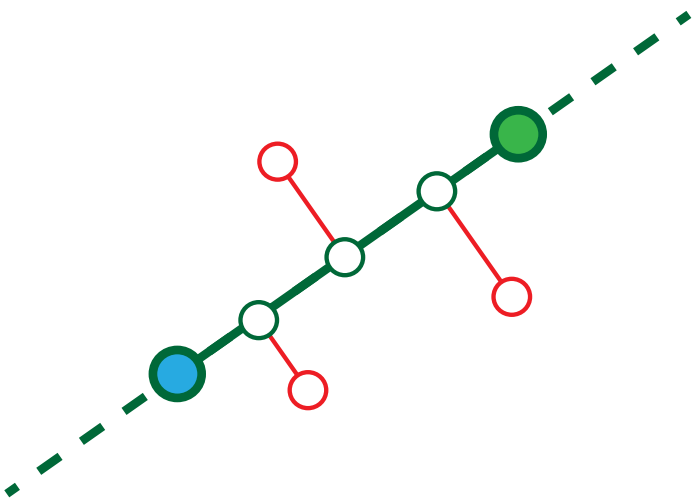

Figure 4.10 – A metric, such as the cosine similarity, can be used to measure the distance between the latent codes obtained by the encoder as-is and the latent codes obtained by interpolating between the first and last latent codes.

Note that in Equation 4.6, we do not compute the cosine similarity for the latent codes of the first and last segments, as $e_0(x) = f_0(x)$ and $e_{N-1}(x) = f_{N-1}(x)$.

We end up with a total of seven possible metrics to compute an anomaly score: the MAE, the MAE with interpolation (MAE-i), the MSE, the MSE with interpolation (MSE-i), the PSNR, the SSIM and the CS.

### 4.2.5 Experimental protocol

**Pre-processing.** Before we proceed to train a model, we pre-process the data.

First, we resize all videos in the dataset to a $128x128$ resolution. For color videos, we additionally convert them to grayscale.

Second, we extract videos of 32 frames. This yields videos with a total of 524288 pixels and 1 color channel.

During training, these videos are simply sampled from training videos. During evaluation, these videos of 32 frames are obtained by sliding a window from start to finish on all testing videos. We do not pad these videos, so from each testing video, we extract $video_{length} - 31$ samples.

Finally, we standardize the extracted videos: the mean is set to 0 and the standard deviation to 1.

**Subsets.** For all datasets, we split them in three subsets: a training set, a validation set and an evaluation set.

The evaluation set of a dataset is the one defined by the authors of this dataset. The training set and validation set are obtained by further splitting the training set defined by the authors of the dataset. The validation set is a small subset of the original training set and is used to assess when the model beings overfitting on the training data.

The validation set contains 20% of the original training set. Therefore, the resulting training set contains the remaining 80%.

Since we use the definitions of the original authors when it comes to defining the original training set and the evaluation set, the relative amounts of data can be found in the description of said datasets in Section 2.2.1.

**Training.** As said above, during training we only use videos from the training set as defined by the authors of the dataset.

For each training step, we add noise to the samples. The noise is taken from a normal distribution centered on zero and with a small variance. Then, the samples are encoded, the codes are interpolated and are then decoded. The reconstruction error is then computed and minimized.

At the end of each epoch, we compare the reconstruction error on the training set to the reconstruction error on the validation set. If the error has been larger on the validation set than the reconstruction set for the second epoch in a row, we stop the training.

**Evaluation.** As said above, during evaluation, we only use videos from the testing set as defined by the authors of the dataset.

For each video, we compute its anomaly score as detailed in Section 4.2.4. Using the ground truth labels provided by the dataset, we flag a video as anomalous if the central image of the video — the 16-th — is flagged as anomalous.

Then, using the anomaly scores and the video labels, we compute the 4 different performance metrics we mentioned in Section 4.2.4.

### 4.2.6 Results

The IAE is structurally very similar to a basic autoencoder. For this reason, one of the focus of our experiments is to assess its contribution in the performance results. In order to do that, we performed an ablation study to compare an IAE and a basic autoencoder in otherwise identical conditions.

Firstly, we evaluated the IAE and the seven proposed metrics for computing the anomaly score on six datasets: the UCSD Ped1, UCSD Ped2, Subway Entrance, Subway Exit, CUHK Avenue and Shanghaitech Campus datasets. We presented all six datasets in Section 2.2.1.

For each dataset and each proposed metric for the anomaly score, we report the values of four performance metrics: the area under the ROC curve, the EER of the ROC curve, the area under the PR curve and the average precision. We presented all four performance metrics in Section 3.3.

We think it is important to make this thorough evaluation for the following reasons:

1. Having multiple ways to compute the anomaly score allows to be sure we are using the best one available or that we need to change the anomaly score metric we are using.
2. Evaluating the IAE on six video surveillance datasets demonstrates it is able to generalize and that the reported results are representative of the overall method's performances.
3. Reporting the results with four performance metrics will allow future works, if they use the same metrics, to compare themselves in a more robust way as it was usually done in the literature.
4. An increased number of performance metrics will help any future potential user to better select the method according to their needs.

We report these results in Table 4.1 (UCSD Pedestrian datasets), Table 4.2 (Subway datasets) and Table 4.3 (CUHK Avenue and Shanghaitech Campus datasets).

| | Metric | ROC | EER | PR | AP |
|---|---|---|---|---|---|
| Ped 1 | MSE | 0.773 | 0.273 | 0.825 | 0.824 |
| | MAE | 0.746 | 0.316 | 0.803 | 0.802 |
| | MSE-i | 0.775 | 0.301 | 0.832 | 0.831 |
| | MAE-i | 0.753 | 0.312 | 0.813 | 0.813 |
| | SSIM | 0.725 | 0.346 | 0.791 | 0.790 |
| | PSNR | 0.788 | **0.272** | 0.836 | 0.835 |
| | CS | **0.790** | 0.290 | **0.854** | **0.852** |
| Ped 2 | MSE | 0.873 | 0.236 | 0.984 | 0.972 |
| | MAE | **0.933** | **0.165** | **0.993** | **0.982** |
| | MSE-i | 0.869 | 0.209 | 0.974 | 0.973 |
| | MAE-i | 0.877 | 0.203 | 0.977 | 0.975 |
| | SSIM | 0.873 | 0.167 | 0.986 | 0.976 |
| | PSNR | 0.879 | 0.243 | 0.985 | 0.972 |
| | CS | 0.898 | 0.186 | 0.978 | 0.977 |

Table 4.1 – Metrics comparison for the UCSD Pedestrian datasets.

| | Metric | ROC | EER | PR | AP |
|---|---|---|---|---|---|
| Entrance | MSE | 0.791 | 0.261 | 0.554 | 0.551 |
| | MAE | 0.803 | 0.255 | **0.582** | **0.580** |
| | MSE-i | 0.798 | 0.268 | 0.540 | 0.536 |
| | MAE-i | **0.806** | **0.245** | 0.558 | 0.556 |
| | SSIM | 0.802 | 0.275 | 0.564 | 0.562 |
| | PSNR | 0.792 | 0.259 | 0.564 | 0.562 |
| | CS | 0.725 | 0.313 | 0.403 | 0.400 |
| Exit | MSE | 0.923 | 0.169 | 0.399 | 0.394 |
| | MAE | **0.932** | **0.144** | 0.439 | 0.433 |
| | MSE-i | 0.908 | 0.167 | 0.231 | 0.229 |
| | MAE-i | 0.919 | 0.164 | 0.318 | 0.317 |
| | SSIM | 0.916 | 0.153 | **0.468** | **0.462** |
| | PSNR | 0.924 | 0.168 | 0.402 | 0.394 |
| | CS | 0.919 | 0.161 | 0.277 | 0.268 |

Table 4.2 – Metrics comparison for the Subway datasets.

| | Metric | ROC | EER | PR | AP |
|---|---|---|---|---|---|
| Avenue | MSE | 0.812 | 0.255 | 0.598 | 0.597 |
| | MAE | 0.799 | 0.284 | 0.607 | 0.605 |
| | MSE-i | 0.796 | 0.286 | 0.577 | 0.575 |
| | MAE-i | 0.777 | 0.310 | 0.566 | 0.564 |
| | SSIM | 0.723 | 0.336 | 0.508 | 0.506 |
| | PSNR | **0.823** | **0.254** | **0.618** | **0.615** |
| | CS | 0.755 | 0.318 | 0.517 | 0.514 |
| ShanghaiTech | MSE | 0.692 | 0.353 | 0.629 | 0.628 |
| | MAE | 0.678 | 0.360 | 0.609 | 0.608 |
| | MSE-i | 0.688 | 0.364 | 0.620 | 0.619 |
| | MAE-i | 0.674 | 0.367 | 0.596 | 0.595 |
| | SSIM | **0.711** | **0.341** | **0.646** | **0.644** |
| | PSNR | 0.703 | 0.349 | **0.646** | **0.644** |
| | CS | 0.688 | 0.365 | 0.617 | 0.601 |

Table 4.3 – Metrics comparison for the CUHK Avenue and ShanghaiTech Campus datasets.

Secondly, while we described the architecture common to all IAEs, instances of the IAE have their own architecture. Since we are doing deep learning, the number of layers, the width of these layers, their class and their implementation all play an important role in the final results.

In order to make sure these specificities were not the sole reason our method performed well, we did an ablation study where we removed the interpolation from training, basically training our model as a basic Autoencoder (AE). All other aspects of the model, training and evaluation were kept the same. We report the results of this comparison made on the UCSD Ped2 dataset in Table 4.4.

| Method | ROC | EER | PR | AP |
|---|---|---|---|---|
| AE | 0.865 | 0.226 | 0.974 | 0.971 |
| IAE | **0.933** | **0.165** | **0.993** | **0.982** |

Table 4.4 – Ablation study of the interpolation during training on the UCSD Ped2 dataset using the MSE to compute anomaly scores.

Thirdly, we compared the IAE with previous works. We report the compared values

| Method | Ped1 | Ped2 | Avenue | ST[1] | Exit | Entrance |
|---|---|---|---|---|---|---|
| Hinami, Mei, and Satoh 2017 | | 0.922 | | | | |
| Y. Zhao et al. 2017 | **0.923** | 0.912 | 0.771 | | | |
| Chong and Tay 2017 | 0.899 | 0.874 | 0.803 | | **0.940** | **0.847** |
| Abati et al. 2019 | | **0.954** | | **0.725** | | |
| Ours | 0.790 | 0.933 | **0.823** | 0.714 | 0.932 | 0.806 |
| Hasan et al. 2016 | 0.810 | 0.900 | 0.702 | | 0.807 | **0.943** |
| W. Liu et al. 2018 | 0.831 | 0.954 | **0.849** | 0.728 | | |
| Luo, W. Liu, and Gao 2017 | | 0.922 | 0.817 | 0.680 | | |
| Feng, Yuan, and X. Lu 2017 | 0.925 | | 0.630 | | | |
| Fan et al. 2018 | 0.949 | 0.922 | | | | |
| Sabokrou, Fayyaz, et al. 2018 | 0.902 | | | | 0.904 | |
| Ravanbakhsh, Nabi, et al. 2018 | 0.957 | 0.884 | | | | |
| Ravanbakhsh, Sangineto, et al. 2019 | **0.968** | **0.955** | | | | |
| Rodrigues et al. 2019 | | 0.922 | 0.829 | **0.760** | | |
| Morais et al. 2019 | | | | 0.734 | | |
| Ours | 0.790 | 0.933 | 0.823 | 0.714 | 0.932 | 0.806 |

[1]ShanghaiTech.

Table 4.5 – Results (ROC) on all datasets. Top: comparison with end-to-end methods. Bottom: comparison with other methods.

in Table 4.5, where the values for previous works were not replicated but taken from their respective publications. Since only the area under the ROC curve was reported for all previous works, we only compare using this value. Note that we split the table in two, putting the end-to-end methods in the top section and other methods at the bottom. This is because non end-to-end methods cannot be transferred to other modalities and often have an edge on end-to-end methods because of their specialization.

Lastly, we compared three different annotations for the Subway Entrance dataset. As we pointed out in Section 2.3.1.2, we think the two existing annotations available are either too narrow or too broad compared to the original definition. We introduced new labels for this dataset and evaluated the IAE on the three sets of annotation. We report the performance results of the IAE in Table 4.6. Our newly introduced labels are available here in the online repository. The given timestamps are tuples of indices of anomalous frames, where the first value indicates the start of an anomalous segment and the second value indicates its end.

| Labels | ROC | EER | PR | AP |
|---|---|---|---|---|
| OriginalAdam et al. 2008 | 0.855 | 0.227 | 0.140 | 0.140 |
| Kim *et al.*J. Kim and Grauman 2009 | 0.806 | 0.245 | 0.558 | 0.556 |
| Ours | 0.834 | 0.230 | 0.204 | 0.203 |

Table 4.6 – Comparison of the IAE's performances depending on the labels used.

### 4.2.7 Discussion

We only tested the IAE on video surveillance data, so there is still work to completely demonstrate the possibility to generalize this architecture to other types of data. As we will see later, we evaluated an extension of the IAE on augmented network packets and show we were able to reach good performance on this modality as well.

We think the IAE is mostly well suited for short anomalies, because of the small length of the input sequence. While this not something specific to IAEs, this problem might impact their performances more significantly as their ability to interpolate latent codes would probably be reduced as well.

However, compared to some previous methods, the IAE is conceptually very simple. This makes it easier to implement, train and test an IAE, making development and tuning quicker as well. The fact an IAE can be used as a simple autoencoder opens up possibilities as well, like the one the IAE was originally designed for: as a feature extractor for another model such as a transformer.

Overall, we showed the IAE is a significant improvement over the basic autoencoder and that it can generalize well. We also showed it performs well compared to previous state-of-the-art methods.

It is worth noting that many previous works were using a basic autoencoder as part of their pipeline. We think there is a good chance the IAE could improve these previous methods, by replacing their autoencoder with an IAE.

### 4.2.8 Future works

As we have hinted before, the IAE should be tested on more modalities, such as audio and it should be evaluated on multimodal data.

Moreover, the IAE currently aims at minimizing the reconstruction error in the input space. The interpolation of latent codes during training constrains the model to produce latent codes lying on straight lines but this is done only in a indirect manner.

It is worth noting that self-supervised learning is also a possibility to move of all this

evaluation process from the input/output space into the latent space. Indeed, the encoder could be trained to encode the different elements of a sequence into points a straight line with a fixed length. To start with, the first and last elements of the sequence would draw this straight line. Then, the objective of the model would be to minimize the distance between the latent codes of the remaining elements and this straight line. Additionally, to prevent the model from collapsing, the encoder would be constrained to have the first and last latent codes to be separated by a fixed distance.

This would directly constrain the model to produce linearly interpolable latent codes, remove the need for a decoder and hopefully improve anomaly detection in the latent space using distance measures such as the cosine similarity.

Additionally, one of the main weaknesses of the IAE is its lack of probabilistic framework. In its current state, the IAE basically assumes there is only one likely way from point A to point B, which is obviously false. This concern is later addressed by the introduction of a probabilistic version of the IAE, which we will address in a following subsection of this chapter.

Finally, since the IAE was originally designed to improve the training of an autoregressive model, such as a transformer, it would only make sense to try if this method was successful for this task as well. It is also worth noting that this could be done in a probabilistic framework, which would help overcome the limitation we just mentioned.

### 4.2.9 Towards multimodality

The inputs for IAEs are split into $N$ segments where each segment has the same length $T$. In the case of multiples modalities, the length of each samples of each modality will have to be divisible by $N \geq 3$ and the total length of each modality will have to be of the same duration.

Once this condition has been met, we can start modeling multiple modalities in a joint manner. We build two encoders and two decoders, one for each modality. Each encoder projects their respective modality into a latent space — one for each modality — and then both resulting latent codes are flattened as explained in Section 4.1.4, if they are not already.

So, let us take a video with audio again, we will call it $x$ where $x_{video}$ and $x_{audio}$ are the video and audio tracks respectively. Using the encoders $e_{video}$ and $e_{audio}$, we get the respective latent codes $z_{video} = e_{video}(x_{video})$ and $z_{audio} = e_{audio}(x_{audio})$. In this case, $z_{video}$ is still a 4-D tensor and $z_{audio}$ is still a 2-D tensor so cannot concatenate them together,

so we flatten them into 1-D tensors with shape $[dim_{video}]$ and $[dim_{audio}]$ respectively. Once concatenated, we obtain a new latent code we will call $z_{joint}$, where $dim_{joint} = dim_{video} + dim_{audio}$.

Then, an additional encoder $e_{joint}$ and an additional decoder $d_{joint}$ are introduced. With this new autoencoder, we can now treat the joint latent code $z_{joint}$ as we treated single modalities. It is encoded, interpolation is performed and it is decoded. At this stage, we split the reconstructed $z_{joint}$ back into $z_{video}$ and $z_{audio}$, reshape them back into 4-D and 2-D tensors respectively, and decode them using their respective decoders.

We are now back into the input space, meaning we can compute the reconstruction error, minimize it during training and use it as the anomaly score during anomaly detection. Using this setup, we should be able to perform joint multimodal anomaly detection.

We illustrate this example in Figure 4.11.

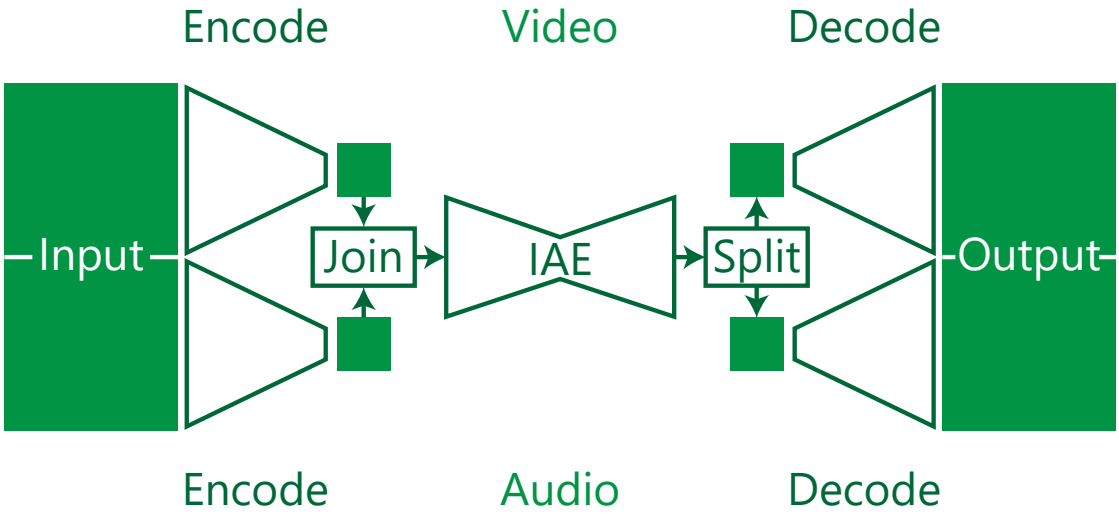


Figure 4.11 – Audio and video are encoded separately, then joined together to be fed to an IAE. The output of the IAE is then split, yielding a code for the video and a code for the audio. These codes are then decoded separately. This setup trains all encoders and decoders in a joint manner and the presence of the bottleneck in the IAE encourages the overall model to draw parallels between the two modalities.

## 4.3 Variational Interpolating Autoencoder

The Variational Interpolating Autoencoder (VIAE) is a method developed for anomaly detection in time series and has been evaluated on the Kitsune dataset, specifically designed to evaluate Network Intrusion Detection (NID) systems.

### 4.3.1 Motivation

We developed the VIAE to overcome one of the main limitations of the IAE: its inability to model multiple likely scenarios at once. As pointed out by Mathieu, Couprie, and LeCun 2015, when using a loss such as the MSE for tasks such as predicting future frames using a deterministic model such as an autoencoder, the model will learn to produce blurry predictions in some cases. These cases happen when the output of the model has several equally likely values. For example, given two equally likely values $v_1$ and $v_2$, the value that minimizes the MSE is $v_{average} = \frac{v_1+v_2}{2}$, even if $p(v_{average})$ itself is very low. We illustrate this phenomenon in Figure 4.12.

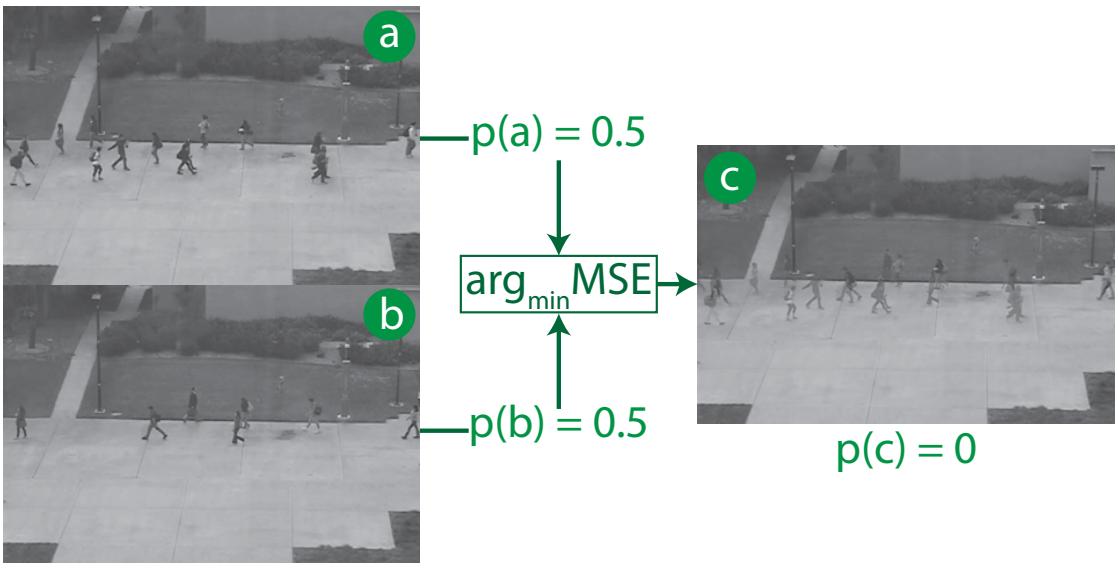


Figure 4.12 – In this example, we assume we start from a situation like an empty street and we want to predict what comes next. We also assume the two presented scenarios a and b are equally likely to be what comes next. For a deterministic autoencoder trained to predict either a or b, the solution that minimizes the prediction error is c, which is extremely unlikely to be observed.

One way to improve the method then is to introduce a probabilistic framework, hence the development of the VIAE.

### 4.3.2 Architecture

As we described it earlier in Section 3.2.1.2, contrary to an AE, a VAE does not directly encode the input into a single latent code $z$, but rather into a latent representation $\mathcal{N}(\mu, \sigma)$ from which any number of latent codes can be sampled: $z \sim \mathcal{N}(\mu, \sigma)$, as shown in Figure 4.13.

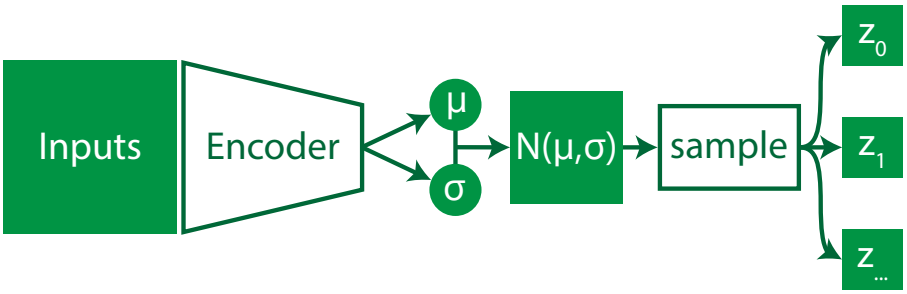


Figure 4.13 – The encoder basically produces two tensors: the mean $\mu$ and the variance $\sigma$ of a normal distribution $\mathcal{N}(\mu, \sigma)$. This normal distribution can be sampled to produce as many latent codes $z_n$ as needed.

### 4.3.3 Objective function and training

As with the IAE, inputs $x$ are split into $N$ segments $x_n$ and then encoded by an encoder $e$. However, contrary to IAEs, $e_n(x) = \{\mu_n, \sigma_n\}$ where $\mu_n$ is the mean and $\sigma_n$ the log of the variance of the posterior distribution $p(z|x_n) = \mathcal{N}(\mu_n, e^{\frac{\sigma_n}{2}})$ from which $z_n$ will be sampled.

In order to sample from $p(z|x_n)$, we will use the reparametrization trick introduced by Kingma and Welling 2013 for VAEs. This reparametrization trick is necessary to keep the model differentiable. Instead of directly sampling from $p(z|x_n)$, we will first sample from a prior distribution $p(z) = \mathcal{N}(0, 1)$. Then, we obtain $z_n$ with the following equation:

$$z_n = \mu_n + e^{\frac{\sigma_n}{2}} \odot \epsilon \qquad \text{where } \epsilon \sim \mathcal{N}(0, 1) \tag{4.7}$$

Once $z_n$ has been sampled, it is decoded in the same way as IAEs, yielding the same

reconstruction error during training: $\mathcal{L}_{IAE}$.

However, since we are using a VAE, we use an additional term to our objective function to regularize the encoder: the Kullback-Leibler Divergence (KLD). This term encourages the model to produce latent distributions similar to $p(z)$ by minimizing the divergence between these distributions.

The KLD is given by:

$$\mathcal{L}_{KLD} = D_{KL}(p(z|x)||p(z)) \tag{4.8}$$

However, since $p(z) = \mathcal{N}(0, 1)$, we can simplify this equation to:

$$\mathcal{L}_{KLD^*} = \frac{\mu_n^2 + e^{\sigma_n} - 1 - \sigma_n}{2} \tag{4.9}$$

This equation also reveals the reason why we are using the log of the variance and not the variance itself. While this slightly complicates things in the reparametrization trick with the addition of an exponential, this removes a potential logarithm operation in the loss. Because the logarithm is only defined for strictly positive values, this prevents errors that would have happened when the encoder would have produced negative values.

Finally, let us introduce a weighting parameter $\lambda$. The parameter $\lambda$ allows weighting between the reconstruction term and the KLD regularization term. The following equation gives us the objective function the model is trained to minimize:

$$\mathcal{L}_{VIAE} = \mathcal{L}_{IAE} + \lambda\mathcal{L}_{KLD^*} \tag{4.10}$$

### 4.3.4 Anomaly detection

The anomaly detection process happens in a very similar way as with IAEs. However, this time we take advantage of the probabilistic framework which gives us the possibility to sample the latent distribution.

The basic idea is as follows: the model can now outputs multiple scenarios but we do not know which one is the most correct one. Let us say for example we are in the situation previously depicted in Figure 4.12, if the model is well fit, it should produce each mode with equal probabilities.

Because of this, we assume that the output with the smallest prediction error will be the output we have to take into consideration. Still using the example in Figure 4.12, if the scenario actually observed is the scenario a, then the model will likely produce a smaller

prediction error when this will be the sampled scenario instead of the other scenario b.

However, we will not be facing situations where we have only two possible scenarios very often and they will even less often be equally likely. This is why we sampled the latent space $K$ times, as shown in Figure 4.14, where $K$ should be big enough to ensure we are modeling enough scenarios but small enough to preserve computational efficiency. In our experiments, we did not try to tune $K$ and always used $K = 64$.

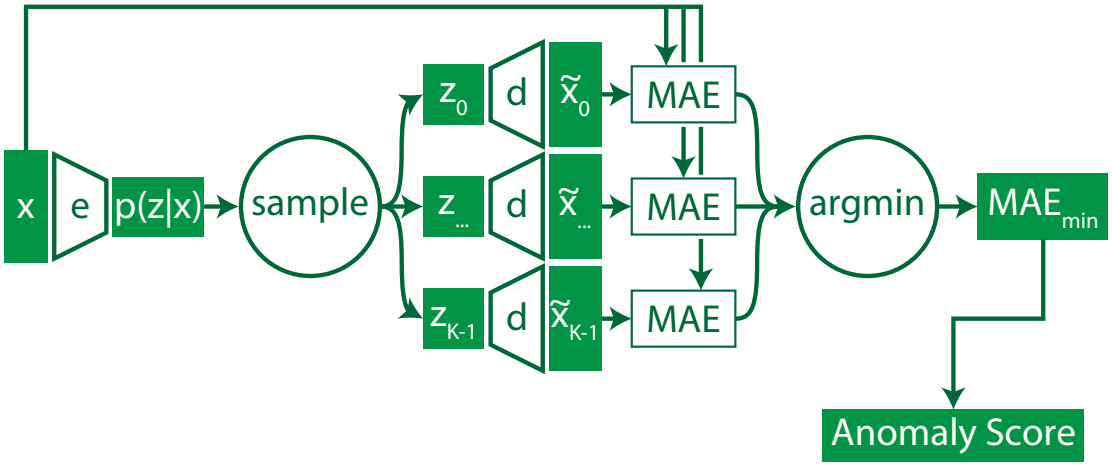


Figure 4.14 – For a given sample $x$, VIAEs yield an anomaly score in four steps. First, the sample is encoded. This gives the posterior distribution of $z$ conditioned on $x$: $p(z|x)$. Second, this distribution $p(z|x)$ is sampled $K$ times. This gives $K$ latent codes $z_k$. Third, the $K$ latent codes are decoded. This gives $K$ outputs $\tilde{x}_k$. And fourth, the MAE is computed for each of the $K$ outputs and the lowest MAE is selected as the anomaly score.

### 4.3.5 Experimental protocol

**Pre-processing.** Before we proceed to train a model, we pre-process the data, similarly as we did with the IAE.

First, we extract sequences of 64 network packets. This yields sequences with a total of 7360 features.

During training, these sequences are simply sampled from the segments reserved for training. During evaluation, these sequences of 64 frames are obtained by sliding a window from start to finish on all testing sequences.

Finally, we normalize the extracted sequences between 0 and 1.

**Training / Validation / Evaluation subsets.** For all 9 distinct attack situations found in the Kitsune dataset, we split them in three subsets as we did with the IAE: a training set, a validation set and an evaluation set.

The evaluation set for a given attack situation is the one defined by the authors of the Kitsune dataset. The training set and validation set are obtained by further splitting the training set as defined by the authors. The validation set is a small subset of the original training set and is used to assess when the model beings overfitting on the training data.

The validation set contains 10% of the original training set. Therefore, the resulting training set contains the remaining 90%.

Since we use the definition of the original authors when it comes to defining the original training set and the evaluation set, the relative amounts of data can be found in the description of the Kitsune dataset in Section 2.2.2.2.

**Training.** As said above, during training we only use sequences from the training set as defined by the authors of the dataset.

For each training step, we add noise to the samples. The noise is taken from a normal distribution centered on zero and with a small variance. Then, the samples are encoded to normal distributions. These normal distributions are then sampled, yielding the latent codes. Then, these codes are interpolated and decoded. Finally, the reconstruction error is computed and minimized alongside the KLD.

At the end of each epoch, we compare the reconstruction error on the training set to the reconstruction error on the validation set. If the error has been larger on the validation set than the reconstruction set for the second epoch in a row, we stop the training.

**Evaluation.** During evaluation, we only use sequences from the testing set, as we did for the IAE.

For each sequence, we compute its anomaly score as detailed in Section 4.3.4. Using the ground truth labels provided by the dataset, we flag a sequence as anomalous if any of its network packets is flagged as anomalous.

Then, using the anomaly scores and the video labels, we compute the 3 different performance metrics (ROC, EER and PR).

### 4.3.6 Results

We now present the results of our experiments with the VIAE on the Kitsune Network Attack Dataset.

Our first set of experiments is aimed at selecting a set of hyper-parameters for our models without relying on the performances results themselves in order to avoid over-fitting. In order to do that, we designed a decision rule based on the one developed by Marteau 2021. We describe this decision rule before presenting its related results.

Our second set of experiments evaluates the model selected by the previous set of experiments and compares this model with previous works in the same conditions.

Our third and last set of experiments is an ablation study where the goal is to assess: 1) the contribution of the probabilistic framework of the VIAE compared to the IAE and 2) the contribution of interpolation in VIAEs compared to VAEs.

#### 4.3.6.1 VIAE — Hyper-parameter selection

The decision rule works as follows:

1. Anomaly scores are computed for all samples.
2. Anomalies are then shuffled and split into 5 partitions of equal sizes.
3. For each partition, only the top 5% biggest anomaly scores are kept.
4. The top 5% anomaly scores of each partition are then compared to those of other partitions, using the bivariate Pearson Correlation.
5. This is done for each set of hyper-parameters and the set that maximizes this correlation value is selected.

In Table 4.7, we present the results of our decision rule for hyper-parameter selection. Each configuration is defined by $c_{size}$, the size of the autoencoder's bottleneck. Values are given for $c_{size} \in \{1, 2, 4, 8, 16, 32\}$ and all 9 situations of the Kitsune Network Attack Dataset.

Then, in Tables 4.8, 4.9 and 4.10 we present the performance results associated with the decision rule values given in Table 4.7.

Finally, in Table 4.11, we regroup the performance results of the configurations which were selected by our decision rule.

| $c_{size}$ | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.965 | **0.982** | **0.991** | 0.915 | **0.985** | 0.969 | 0.983 | **0.994** | 0.940 |
| 2 | 0.992 | 0.980 | 0.988 | 0.978 | 0.954 | 0.955 | 0.983 | 0.983 | 0.973 |
| 4 | 0.989 | 0.964 | 0.988 | 0.915 | 0.962 | 0.992 | **0.993** | 0.983 | 0.970 |
| 8 | **0.995** | 0.967 | 0.986 | **0.989** | 0.960 | 0.916 | 0.985 | 0.964 | 0.988 |
| 16 | 0.988 | 0.969 | 0.987 | 0.984 | 0.961 | **0.996** | 0.981 | 0.992 | 0.986 |
| 32 | 0.991 | 0.956 | 0.986 | 0.969 | 0.969 | 0.995 | 0.977 | 0.990 | **0.994** |

Table 4.7 – Hyper-parameter selection results for all $c_{size}$ and for all situations in the Kitsune Network Attack Dataset. Higher is better.

| $c_{size}$ | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.973 | 0.740 | 0.5 | **1.0** | 0.994 | **1.0** | 0.957 | 0.782 | 0.809 |
| 2 | 0.988 | 0.720 | 0.5 | **1.0** | 0.996 | **1.0** | 0.951 | 0.795 | 0.865 |
| 4 | 0.990 | 0.774 | 0.5 | **1.0** | 0.996 | **1.0** | 0.965 | 0.811 | 0.882 |
| 8 | 0.990 | **0.776** | 0.5 | **1.0** | 0.997 | **1.0** | **0.976** | 0.831 | 0.945 |
| 16 | 0.991 | 0.725 | 0.5 | **1.0** | 0.997 | **1.0** | 0.969 | 0.818 | 0.943 |
| 32 | **0.992** | 0.726 | 0.5 | **1.0** | **0.998** | **1.0** | 0.972 | **0.844** | **0.954** |

Table 4.8 – ROC(AUC) values for all $c_{size}$ and for all situations in the Kitsune Network Attack Dataset. Higher is better. Values were rounded to three decimal places.

| $c_{size}$ | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.082 | 0.328 | 0.5 | 0.003 | 0.024 | **0.0** | 0.102 | 0.295 | 0.269 |
| 2 | 0.040 | 0.342 | 0.5 | **0.001** | 0.018 | **0.0** | 0.112 | 0.289 | 0.220 |
| 4 | 0.033 | 0.301 | 0.5 | 0.003 | 0.018 | **0.0** | 0.105 | 0.275 | 0.208 |
| 8 | 0.034 | **0.296** | 0.5 | **0.001** | 0.018 | **0.0** | **0.081** | 0.262 | 0.136 |
| 16 | 0.030 | 0.340 | 0.5 | **0.001** | 0.017 | **0.0** | 0.095 | 0.274 | 0.135 |
| 32 | **0.029** | 0.335 | 0.5 | **0.001** | **0.015** | **0.0** | 0.091 | **0.247** | **0.118** |

Table 4.9 – EER values for all $c_{size}$ and for all situations in the Kitsune Network Attack Dataset. Lower is better. Values were rounded to three decimal places.

| $c_{size}$ | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.992 | 0.909 | 0.760 | **1.0** | 0.990 | **1.0** | 0.962 | 0.148 | 0.507 |
| 2 | 0.997 | 0.902 | 0.751 | **1.0** | 0.993 | **1.0** | 0.957 | 0.180 | 0.655 |
| 4 | 0.997 | 0.921 | 0.754 | **1.0** | 0.993 | **1.0** | 0.967 | 0.194 | 0.705 |
| 8 | 0.997 | **0.924** | 0.755 | **1.0** | 0.993 | **1.0** | **0.976** | 0.213 | 0.861 |
| 16 | **0.998** | 0.900 | 0.756 | **1.0** | 0.994 | **1.0** | 0.970 | 0.243 | 0.846 |
| 32 | **0.998** | 0.895 | **0.767** | **1.0** | **0.995** | **1.0** | 0.972 | **0.271** | **0.870** |

Table 4.10 – PR(AUC) values for all $c_{size}$ and for all situations in the Kitsune Network Attack Dataset. Higher is better. Values were rounded to three decimal places.

| | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| $c_{size}$ | 8 | 1 | 1 | 8 | 1 | 16 | 4 | 1 | 32 |
| ROC | 0.990 | 0.740 | 0.5 | 1.0 | 0.994 | 1.0 | 0.965 | 0.782 | 0.954 |
| EER | 0.034 | 0.328 | 0.5 | 0.001 | 0.024 | 0.0 | 0.105 | 0.295 | 0.118 |
| PR | 0.997 | 0.909 | 0.760 | 1.0 | 0.990 | 1.0 | 0.967 | 0.148 | 0.870 |

Table 4.11 – Selected values of $c_{size}$ and their respective performance results. Values were rounded to three decimal places.

#### 4.3.6.2 VIAE — Comparison with previous methods

In Tables 4.12, 4.13 and 4.14 we compare the performance results of VIAEs with those of previous works on the Kitsune dataset.

In all following tables, we include both the performance results of the selected and the best performing configuration of our model.

We also include two sets of results for KitNet: 1) Values from the first set, labeled KitNet, are directly taken from the KitNet paper (Mirsky et al. 2018). 2) Values from the second set, labeled KitNet*, are obtained by running the available code in the associated GitHub repository.

Since values for the area under the PR curve were not available in the KitNet paper, we only report them for the second set.

Finally, we include values for the area under the ROC and PR curves and for the following methods: Isolation Forests (IF) (F. T. Liu, Ting, and Z.-H. Zhou 2008), Extended Isolation Forests (EIF) (Hariri, Carrasco Kind, and Brunner 2019), DiFF-RF (point-wise) and DiFF-RF (collective) (Marteau 2021). These values were obtained from the DiFF-RF paper (Marteau 2021).

| Method | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| KitNet | 0.909 | **0.795** | **1.0** | 0.999 | 0.948 | **1.0** | 0.974 | **0.950** | 0.854 |
| KitNet* | 0.806 | 0.662 | 0.829 | 0.945 | 0.900 | **1.0** | 0.983 | 0.723 | 0.952 |
| IF | 0.532 | 0.5 | 0.5 | 0.953 | 0.927 | 0.999 | 0.994 | 0.641 | 0.801 |
| EIF | 0.5 | 0.5 | 0.5 | 0.893 | 0.897 | 0.999 | 0.973 | 0.624 | 0.655 |
| DiFF-RF (point-wise) | 0.770 | 0.591 | 0.5 | 0.991 | 0.936 | 0.999 | **0.998** | 0.744 | 0.947 |
| DiFF-RF (collective) | 0.845 | 0.5 | 0.995 | 0.983 | 0.936 | 0.999 | 0.968 | 0.796 | 0.905 |
| VIAE (selected) | 0.990 | 0.740 | 0.5 | **1.0** | 0.994 | **1.0** | 0.965 | 0.782 | **0.954** |
| VIAE (best) | **0.992** | 0.776 | 0.5 | **1.0** | **0.998** | **1.0** | 0.976 | 0.844 | **0.954** |

Table 4.12 – Comparison of ROC(AUC) values with previous works. Higher is better. Values were rounded to three decimal places.

| Method | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| KitNet | 0.173 | 0.452 | **0.001** | 0.003 | 0.094 | 0.00104 | 0.108 | **0.136** | 0.268 |
| KitNet* | 0.241 | 0.367 | 0.239 | 0.103 | 0.167 | 0.00327 | **0.025** | 0.421 | **0.087** |
| VIAE (selected) | 0.034 | 0.328 | 0.5 | **0.001** | 0.024 | 0.00022 | 0.105 | 0.295 | 0.118 |
| VIAE (best) | **0.029** | **0.296** | 0.5 | **0.001** | **0.015** | **0.00020** | 0.081 | 0.247 | 0.118 |

Table 4.13 – Comparison of EER values with previous works. Lower is better. Values were rounded to three decimal places.

| Method | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| KitNet* | 0.885 | 0.853 | 0.783 | 0.998 | 0.503 | 0.999 | 0.845 | 0.042 | **0.939** |
| IF | 0.729 | 0.719 | 0.431 | 0.998 | 0.578 | 0.999 | 0.955 | 0.246 | 0.572 |
| EIF | 0.718 | 0.694 | 0.442 | 0.989 | 0.494 | 0.997 | 0.795 | 0.255 | 0.297 |
| DiFF-RF (point-wise) | 0.904 | 0.845 | 0.547 | **1.0** | 0.625 | 0.999 | **0.992** | 0.243 | 0.910 |
| DiFF-RF (collective) | 0.924 | 0.817 | **0.988** | 0.999 | 0.625 | 0.999 | 0.962 | 0.250 | 0.850 |
| VIAE (selected) | 0.997 | 0.909 | 0.760 | **1.0** | 0.990 | **1.0** | 0.967 | 0.148 | 0.870 |
| VIAE (best) | **0.998** | **0.924** | 0.767 | **1.0** | **0.995** | **1.0** | 0.976 | **0.271** | 0.870 |

Table 4.14 – Comparison of PR(AUC) values with previous works. Higher is better. Values were rounded to three decimal places.

#### 4.3.6.3 VIAE — Ablation Study

In Table 4.15, we present values for the area under the ROC curve for our ablation study. We compare the VIAE to both the VAE and the IAE. To ensure a fair comparison, both the VAEs and the IAEs were trained using the same configurations as those selected for VIAEs by our decision rule for meta-parameter selection.

| Method | Active Wiretap | ARP MitM | Fuzzing | Mirai Botnet | OS Scan | SSDP Flood | SSL R. | SYN DoS | Video Injection |
|---|---|---|---|---|---|---|---|---|---|
| VAE | 0.977 | 0.591 | 0.5 | **1.0** | 0.997 | **1.0** | 0.961 | 0.716 | 0.960 |
| IAE | 0.990 | 0.734 | 0.5 | **1.0** | 0.996 | **1.0** | 0.975 | 0.829 | **0.963** |
| VIAE (selected) | 0.990 | 0.740 | 0.5 | **1.0** | 0.994 | **1.0** | 0.965 | 0.782 | 0.954 |
| VIAE (best) | **0.992** | **0.776** | 0.5 | **1.0** | **0.998** | **1.0** | **0.976** | **0.844** | 0.954 |

Table 4.15 – Comparison of ROC(AUC) values when disabling individual components of VIAEs. Higher is better. Values were rounded to three decimal places.

### 4.3.7 Discussion

Every packet of a network stream has been generated by a latent process. Normal processes, which are temporally coherent (i.e. packets are partially dependant in a subsequence), generate most of the observed data. In NID systems, the goal is to detect samples which were generated by a subset of anomalous processes, whose very nature is to aim at disturbing the proper functioning of network.

Individual packets in the Kitsune Network Attack Dataset already contain some temporal information. This is a first step towards modeling the temporal nature of network streams. We go further in developing a model which 1) aims at modeling underlying temporal processes and 2) takes into account the temporal context. This context captures the local and sequential dependencies of network packets.

The importance of such temporal modeling is notably demonstrated by the results presented in Table 4.15, where is shown that the IAE component is the main contributor to our model's performances.

When looking at Tables 4.12, 4.13 and 4.14, our model performs better than state-of-the-art methods in most situations across all performance metrics and sometimes by a wide margin.

However, on Fuzzing, our model does not perform better than a random binary classifier. This problem is also encountered by other methods, such as IF, EIF and DiFF-RF (point-wise). We believe this problem is caused by the distribution of errors across the 115 features of individual packets, as both KitNet and DiFF-RF (collective) model this distribution in their own way.

### 4.3.8 Future works

First, improvements related to the sampling of the latent distribution could be made. During training, multiple latent codes could be sampled per step, similarly to what is done during the anomaly detection process. In this case, only the smallest error would be considered for training the model. We think this would encourage the model to better separate likely scenarios by not penalizing it when predicting another scenario because the sampling of the latent distribution picked a region corresponding to another scenario. However, this would drastically reduce the training speed of the model.

During anomaly detection, we could still consider the sample with the smallest reconstruction error but instead of using this reconstruction error as the anomaly score, we could use the likelihood of the latent code that led to this score based on the latent distribution or the prior, depending on future works results.

Second, we designed a decision rule for selecting a model that is not based on the performance of said model. However, we discovered afterwards the existence of the ROCCH which allows to combine the ROC curves of multiple models into one. Knowing that, future works probably should build this ROCCH curve and select the model(s) based on this.

For the Kitsune dataset specifically, it would be interesting to investigate further the reasons why multiple methods seem to not be able to detect anomalies on the Fuzzing subset.

While did not try to tune $K$, the number of samples taken from $p(z|x)$, it would be worth investigating the impact of $K$ on the computational cost and the accuracy of the method.

### 4.3.9 Towards multimodality

Building a VIAE for joint multimodal anomaly detection would follow the same steps as the ones we describe for the IAE. However, there is the question of where the sampling process should happen. We believe that, in order to keep models simple and easy to train, the sampling of the latent distribution should only occur in the deepest autoencoder, that is to say, the autoencoder responsible for the joint modeling of modalities.

## 4.4 Clear & Concise

Before we describe the CnC, please note the CnC is still an experimental method. We include it here because it has strong theoretical foundations and it yielded state-of-the-art results on one dataset so far.

### 4.4.1 Motivation

Anomalies can be seen as samples less likely to be observed than normal samples. Starting from there, the CnC aims at estimating the likelihood of a given sample in an information theory inspired manner.

#### 4.4.1.1 Theoretical background

Let us take a random variable $X$ defined as the following triple: $X = (x, A_X, P_X)$, where x is the outcome of $X$, $A_X = \{a_1, a_2, ..., a_I\}$ is the set of possibles values for $x$ and $P_X = \{p_1, p_2, ..., p_I\}$ are the probabilities for each of these values. Information theory describes the Shannon entropy $H(X)$ of this random variable $X$ as follows (MacKay and Mac Kay 2003):

$$H(X) = -\sum_{x \in A_X} P(x) \cdot \log_2 P(x) \tag{4.11}$$

The Shannon entropy $H(X)$ can also be described as the mathematical expectation of the information content $h(x)$ for the outcome $x$, where $h(x)$ is defined by:

$$h(x) = -\log_2(P(x)) \tag{4.12}$$

Both the Shannon entropy $H(X)$ and the information content $h(x)$ are measured in bits.

Now, let us consider $x$ is the observed sample from the context $X$ and $x$ is an anomaly if $P(x) < \epsilon$. Then, the problem becomes to estimate $P(X)$ and fix $\epsilon$ depending on the desired system. However, $P(X)$ is intractable because we do not know the normalizing constant. Computing this normalizing constant would require to enumerate all possible samples in $A_X$ and their respective likelihood, but such enumeration is not possible, notably for reasons we gave in Section 1.2.1.

So, instead of computing $P(X)$ directly, the aim of the CnC method is to estimate the other term in Equation 4.12: the information content $h(x)$.

In the framework of lossless compression, information theory also introduces the notion of source coding and optimal source codelengths. Source coding is the encoding of any outcome $x$ into a variable length vector $c(x)$ using a binary symbol coding table $C$ and $l(x)$ denotes the length of $c(x)$. $C$ is said to be optimal if it minimizes the expected length $L(C, X)$. The expected length $L(C, X)$ is minimized and equal to the Shannon entropy $H(X)$ if and only if the codelengths are equal to the information contents $h(x)$. Assuming $L(C, X) = H(X)$, we get the following equality:

$$l(x) = -\log_2(P(x)) = h(x) \tag{4.13}$$

However, while the Shannon entropy $H(X)$ is the lower bound of $L(C, X)$, it is not always possible to compress as far as $H(X)$. Fortunately, information theory defines an upper bound as well:

$$H(X) \leq L(C, X) < H(X) + 1 \tag{4.14}$$

Additionally, if $C$ is optimal, it satisfies Equation 4.14 and $l(x) \leq \lceil h(x) \rceil$. Therefore, we get:

$$h(x) \leq l(x) \leq \lceil h(x) \rceil < h(x) + 1 \tag{4.15}$$

$$0 \leq l(x) - h(x) < 1 \tag{4.16}$$

$$l(x) - 1 < h(x) \leq l(x) \tag{4.17}$$

Assuming $C$ is an optimal coding table, Equation 4.17 shows the codelength $l(x)$ can be used to bound to the information content $h(x)$ and therefore estimate the likelihood $x$ is anomalous.

#### 4.4.1.2 Practical considerations

In practice, we know neither $C$ nor $l(x)$ and we do not have access to them. However, we can relax the problem by making the following assumption:

We assume a lossy variable length encoder $e$ can be trained to tend towards an optimal coding table $C$ by minimizing both the codelengths and the information lost in the encoding process.

This allows us to model $C$ and $l(x)$ and therefore makes the overall problem tractable.

### 4.4.2 Architecture

We developed several versions of the CnC, each with their own strengths and weaknesses. Here, we will describe the overall architecture that most versions shared.

The CnC currently relies on a variable length autoencoder with three parts: an encoder $e$, a decoder $d$ and a codelength estimator $m$. The encoder and the decoder work as they usually do in autoencoders while the codelength estimator produces and applies a mask on the latent codes produced by the encoder before they are fed to the decoder.

In most versions, $m$ estimates the codelength based on the latent codes $z = e(x)$ where $x$ is the input. Therefore, we have $m(z) = m(e(x))$, implying that the mask depends on the input $x$. The estimated variable-length code $\tilde{z}$ for $x$ is then given by $\tilde{z} = z \odot m(z)$ and the reconstruction $\tilde{x}$ is given by $\tilde{x} = d(\tilde{z})$. This inference process is illustrated in Figure 4.15.

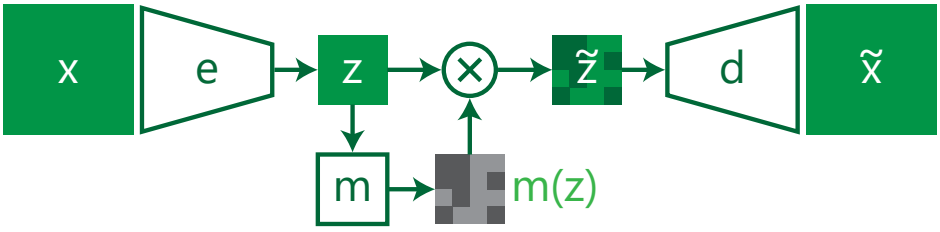


Figure 4.15 – The input $x$ is first encoded by $e$, yielding the fixed size code $z$. Then, the codelength estimator $m$ produces a binary mask that is applied to $z$, yielding the variable length code $\tilde{z}$. Finally, $\tilde{z}$ is decoded by $d$, yielding the output $\tilde{x}$.

Concerning $m$, it is a stack of causal 1D CNNs where the chosen dimension is the dimension of features. This implies that for predicting the mask of the first feature, $m$ only has access to this first feature. When it comes to the second one, $m$ has access to the two first features, and so on. We illustrate this causal relationship in Figure 4.16.

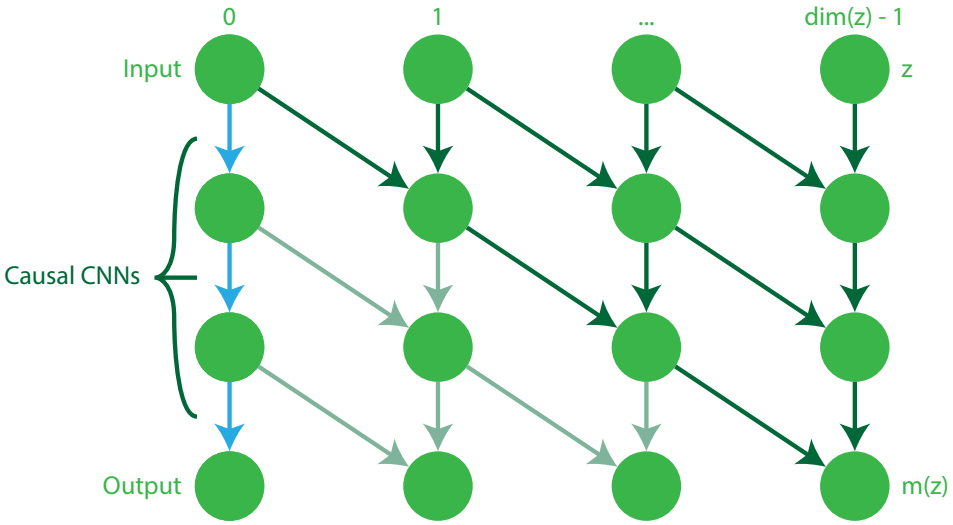


Figure 4.16 – The first output — index 0 — only has access to the nodes linked by blue arrows, the rest are masked out. As the index of the considered output increases, less inputs are masked out. The last output has access to all inputs, as shown by the dark green arrows. The rows of arrows depict the causal CNNs in play.

The motivation behind the usage of causal CNNs is to encourage the encoder $e$ to order its features such that the first feature is unlikely to be masked by $m$, the second feature is slightly more likely to be masked by $m$, and so on.

### 4.4.3 Objective function and training

To start with, we make sure CnCs can be fully differentiated in order to use gradient descent. This is particularly important here because we apply binary masks to the latent codes in order to simulate variable length codes. Then, we define the loss functions CnCs have to minimize. Finally, we add a note on the learning rate for the architecture we described above.

#### 4.4.3.1 Differentiable binary mask

$m$ is a deep learning model, meaning as-is it will produce real values, not binary values. This means we have to perform binarisation of these values. However, binarisation typically is not a differentiable but there are workarounds. We note $\hat{m}(z)$ the real-valued

mask before binarisation, $b : \mathbb{R} \mapsto \{0, 1\}$ a non-differentiable binarisation function and $\not\nabla$ represents the stop-gradient operator. Using $\hat{m}(z)$, $b$ and $\not\nabla$, Equation 4.18 defines a differentiable binarisation process:

$$m(z) = b(\hat{m}(z)) + \hat{m}(z) - \not\nabla(\hat{m}(z)) \tag{4.18}$$

The stop-gradient operator prevents gradients from being backpropagated through the second $\hat{m}(z)$. During inference, the two $\hat{m}(z)$ basically cancel each others, only leaving the binary values from $b(\hat{m}(z))$. However, during backpropagation, gradients only flow through $\hat{m}(z)$, ignoring the two other terms because they are not differentiated. In a way, $\hat{m}(z) - \not\nabla(\hat{m}(z))$ acts as additional noise that happens to only leave zeros and ones after it has been added to $\hat{m}(z)$.

#### 4.4.3.2 Loss terms

As we said above, the goal is to minimize both the average codelength and the average information lost during the encoding process. In our case, this gives us two loss functions to minimize: $\mathcal{L}_{clear}$ and $\mathcal{L}_{concise}$. On one hand, $\mathcal{L}_{clear}$ is responsible for minimizing the average information lost and is formulated as the reconstruction error between the input $x$ and the output $\tilde{x}$ of the autoencoder as shown in Equation 4.19:

$$\mathcal{L}_{clear} = \|\tilde{x} - x\|^2 \tag{4.19}$$

On the other hand, $\mathcal{L}_{concise}$ is responsible for minimizing the average codelength, which is simply the sum of the values of the real-values mask $\hat{m}(z)$. Assuming $e$ produces latent codes $z$ with dimensionality $dim(z)$, Equation 4.20 gives the second loss term $\mathcal{L}_{concise}$:

$$\mathcal{L}_{concise} = \sum_{i=1}^{dim(z)} \hat{m}_i(z) \tag{4.20}$$

Additionally, if $\hat{m}_i(z)$ is not bounded by the architecture, it is clipped in the loss term to prevent the model from ignoring the other loss term $\mathcal{L}_{clear}$, as focusing on minimizing $\mathcal{L}_{concise}$ to $-\infty$ would become a trivial solution for the model.

#### 4.4.3.3 Weighting clarity and conciseness

As with VIAEs in their loss function shown in Equation 4.10, the two different loss terms must be weighted so that the model minimizes the two terms in a balanced man-

ner. For example, a CnC focusing on the $\mathcal{L}_{clear}$ term would simply behave like a basic autoencoder.

For this reason, we introduce a dynamic balancing factor $\lambda_{concise}$ based on the reconstruction loss $\mathcal{L}_{clear}$ and a threshold $\theta_{clear}$. This balancing factor is detailed in Equation 4.21:

$$\lambda_{concise} = \frac{\theta_{clear} - \not\nabla(\mathcal{L}_{clear})}{\theta_{clear}} \tag{4.21}$$

The motivation behind the formulation of this balancing factor is the following: Ideally, the information loss would be constant. The reconstruction error being our proxy for estimating the information loss, the goal is to have the model focus more on reconstruction above a given threshold $\theta_{clear}$. When this term is negative, it encourages the model to produce longer codes, hopefully preventing it from getting stuck with 0-length codes and a high reconstruction error. When the term becomes positive, it encourages the model to start reducing the codelengths.

However, defining $\theta_{clear}$ is not a trivial task. We first started with a constant close to the reconstruction error of a well-trained autoencoder in the same conditions. This mostly led the model to ignore $\mathcal{L}_{concise}$ and focus on reconstruction. At the moment where $\mathcal{L}_{clear}$ becomes less than $\theta_{clear}$, the model is too far gone in the "increase codelengths" direction to ever go back.

Similarly as to J. Zhao, Mathieu, and LeCun 2016 with their margin value, we decided to decay $\theta_{clear}$ over time. In our case, the decay schedule for $\theta_{clear}$ was set to be an exponential decay curve fit on the reconstruction error from previous runs. While this solution is *ad hoc*, this definitely improves the stability between the two loss terms during training.

It is still not clear how $\lambda_{concise}$ should be computed and it remains the one of the main challenges to overcome for the CnC to properly converge and work.

#### 4.4.3.4 Mask-less loss term

We still need one element to obtain our final loss term. Indeed, while we made sure gradients flow through the binarisation for the codelength estimator $m$, it is not always the case for the encoder $e$. When $m$ produces zeros, the gradients for $e$ are multiplied by zero, effectively stopping them. Because $m$ will tend to deactivate the same features first, some of them are never updated. Because these features are not updated, they will

increase the reconstruction error if they were used, encouraging $m$ even more to keep them masked, and so on.

In order to break this loop and allow the encoder to update these features, we add an additional reconstruction loss term $\mathcal{L}_{skip}$ that skips the mask produced by $m$. $\mathcal{L}_{skip}$ is defined by Equation 4.22:

$$\mathcal{L}_{skip} = \|d(e(x)) - x\|^2 \tag{4.22}$$

#### 4.4.3.5 Final loss term

Finally, combining Equations 4.19, 4.20, 4.21 and 4.22 together and an additional constant weight $\lambda_{skip}$ for balancing $\mathcal{L}_{skip}$, the total loss term $\mathcal{L}_{CnC}$ minimized by CnCs is given by Equation 4.23:

$$\mathcal{L}_{CnC} = \mathcal{L}_{clear} + \lambda_{concise}\mathcal{L}_{concise} + \lambda_{skip}\mathcal{L}_{skip} \tag{4.23}$$

#### 4.4.3.6 Learning rate

In the architecture illustrated in Figure 4.15, the codelength estimator $m$ outputs masks based on $z$, which depends on $x$ and importantly $e$, the encoder. However, $z$ is also used for reconstructing $x$ and therefore $e$ is also trained to reconstruct $x$, in addition to assisting $m$ in creating masks. Because of this, it may seem that $m$ is always lagging behind $e$, as $z$ may be very different from one training step to another, ultimately leading in an unstable training where $m$ can never really exploit $z$.

In order to prevent this problem from leading to an unstable training, we used a small learning rate ($10^-3$ typically).

### 4.4.4 Anomaly detection

Once a CnC has been trained, we have three ways to use it to compute anomaly scores. We illustrate these three ways in Figure 4.17 and detail them after.

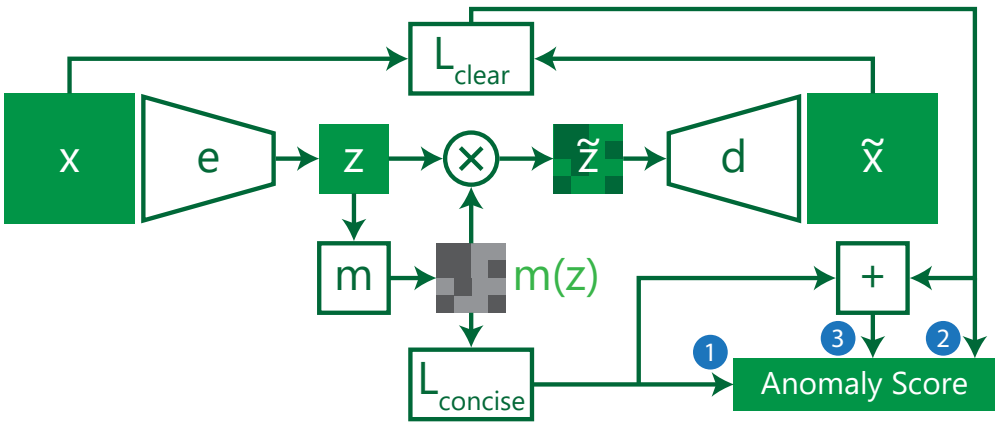


Figure 4.17 – The anomaly score can be computed in three ways. 1) Only using the estimated codelength. 2) Only using the reconstruction error. 3) By individually normalizing 1 and 2 and then adding them.

First, we can use it as we originally intended when looking at the theoretical background: only use the codelengths of our sample. These codelengths should relate to the likelihood of their respective samples. There are two main reasons for using this first solution. The first reason is the reduced computational cost as the decoder is removed and we do not need to compute the difference between the input and the output. The second reason is to avoid making comparison in an input space for which distance measurements do not correlate with perceptual differences. This is particularly the case when human behaviors are being observed through raw modalities such as audio and video.

Second, we can simply use the reconstruction error (without the skip connection). Depending on the input space, the task and how well the training phase went, the reconstruction error can still help in detecting anomalies. The main reason for using the reconstruction error is the fact that, unless the model is very well fit, it will likely still reconstruct some samples better than others.

Third, the two previous values can be combined together. To do so, they are first normalized individually and then added. There are two reasons for using this as the anomaly score. First, the model was not train to solely minimize one of the two loss terms and while the reconstruction loss term mainly exists to previous the codelength loss term to collapse, it is still relevant. Second, this combines the strengths and weaknesses of the two previous solutions and can yield improved performances as suggested by our experiments.

### 4.4.5 Results

Our results are two-fold. The first part concerning the training phase and the second part the anomaly detection phase.

#### 4.4.5.1 Training CnCs

We note that at the moment, if the schedule for $\theta_{clear}$ is not carefully chosen, the model will converge towards an unwanted solution — minimizing only of the two terms — or even diverge. To choose $\theta_{clear}$, we first observed the reconstruction loss of the autoencoder when trained alone. Then, we set $\theta_{clear}$ to be an exponentially decaying curve following closely the observed reconstruction loss. This choice of $\theta_{clear}$ did not work in all situations but seemed to provide stable results more regularly that trial and error. Once $\theta_{clear}$ was correct, the model became stable enough to allow the replication of the results over different runs.

Another important choice to be made when building a CnC is the choice of $b$, the non-differentiable binarisation function. We first used a basic thresholding method, where $\hat{m}(z)$ is mapped to 1 if it is above a given value and to 0 otherwise. However, we found this to significantly reduce the stability of the model. Indeed, while $\hat{m}(x)$ is real-valued, $m(h)$ is not. A slight change in $\hat{m}(x)$ can be enough to create a sudden change in $m(h)$, which is not good for convergence because this creates a sudden steep slope for the loss function.

We moved from using this method to using a probabilistic method, where the threshold is basically sampled from a uniform distribution. While still introducing some sudden changes, this method should benefit more from the stochastic mini-batch training by averaging these changes over the samples in the batch. This change improved stability overall but seemed to reach the same anomaly detection performances as an already stable model.

#### 4.4.5.2 Early anomaly detection results

While the CnC is still an experimental method, we have managed to produce some early good results on the UCSD Ped2 dataset. Although good results on such datasets are not enough to validate the method, this is a step in the right direction.

More importantly, using only the estimated codelengths as anomaly scores, we were able to reach an area under the ROC curve of 0.923 for the UCSD Ped2 dataset. While these are not state-of-the-art results, this indicates our novel way to estimate anomaly

scores based on the codelength — instead of the usual reconstruction error — is a valid approach. We show more results in Table 4.16, where we compare the IAE with the CnC where the anomaly scores are the combination of estimated codelengths and reconstruction errors.

In order to keep the comparison fair, we used the same experimental protocol as we did for the IAE.

| | ROC | EER | PR | AP |
|---|---|---|---|---|
| IAE | 0.933 | 0.165 | **0.993** | 0.982 |
| CnC (combined) | **0.939** | **0.099** | 0.990 | **0.988** |

Table 4.16 – Comparison between the IAE and the CnC (with combined anomaly scores) on the UCSD Ped2 dataset. The same experimental protocol was used for both methods.

### 4.4.6 Discussion

While incomplete, we showed the CnC is a promising method. When compared on the UCSD Ped2 dataset with the IAE, the CnC performs slightly better. Additionally, the estimated codelengths can be used to detect anomalies as originally intended.

The fact that we have to manually define a schedule for $\theta_{clear}$ to get good results is unpractical. Ideally, this value would be set dynamically based on $\mathcal{L}_{clear}$ or the balancing factor $\lambda_{concise}$ would be modified so that $\theta_{clear}$ can be discarded.

### 4.4.7 Future works

We made a few attempts at automatically defining $\theta_{clear}$ instead of a fixed schedule without much success at the moment. Future works should be done to automatically define $\theta_{clear}$, as this would allow generalising to new datasets without having to go through the non trivial process of manually defining $\theta_{clear}$.

After that, there may be some stability and convergence issues left to work on. For all stability and convergence issues, a look back on the theoretical background will probably helpful in solving these issues.

Once the CnC will have reached a satisfying state regarding stability, it will be necessary to evaluate it on other datasets. Contrarily to the IAE and the VIAE, the CnC is not limited to sequential data. Because the CnC was built from a theoretical base on anomalies in general, it could well possibly prove useful on any type of data.

Currently, the CnC is formulated as an autoencoder with an extra module for estimating codelengths. This was motivated by the notion of compression present in the theoretical background and the ability of autoencoders to compress data. However, we only really need an encoder $e$ to model the coding table $C$ and a way to measure or approximate the amount of information lost by the encoder $e$.

This notably opens us the possibility to train the encoder in a self-supervised manner. In particular, this enables context specific ways to estimate the mutual information between the input $x$ and the variable length code $\tilde{z}$. For example, Bachman, Hjelm, and Buchwalter 2019 trained an encoder to maximize mutual information between the codes across different views of the same image. In the case of anomalous human behavior, a similar method could be developed for maximizing the mutual information across different views of the same behavior. For the CnC, this would mean maximizing the mutual information between the different variable length codes across different views of the input.

It is also worth investigating the architecture of the codelength estimator $m$. We used causal 1D CNNs, future works should evaluate the usage of a Recurrent Neural Network (RNN) such as a Long Short-Term Memory (LSTM) or even a simple autoregressive model where the probability $p(m_i(z))$ is conditioned on $p(m_{i-1}(z))$ and $z$: $p(m_i(z)|m_{i-1}(z), z)$.

Additionally, it is worth investigating a setup where $e$ and $m$ are merged into a single sub-model able to produce a "stop" token. Until this token is seen, this sub-model would continue to output features, effectively building the variable length code step by step. Then, we have two solutions to estimate the codelengths. First, we can use the number of steps taken. Second, we can use the sum of the logits used to produce the stop token over the different. The difference in these two solutions is basically the same as the difference between using $\hat{m}_i(z)$ and $m_i(z)$ for computing $\mathcal{L}_{concise}$ in Equation 4.20.

### 4.4.8 Towards multimodality

We see two straightforward ways the CnC could be adapted for joint multimodal anomaly detection.

The first one is to simple use a multimodal autoencoder as illustrated in Figure 4.11, preferably using a perceptual loss for training the autoencoder.

The second way would be to consider the multiple modalities to be views of the same mechanisms. Using only an encoder $e$ and a codelength estimator $m$, the goal would be to maximize the mutual information between the different views and modalities. Additional views for each modality would probably be helpful as the information on a given behavior

might not be present in all modalities (*e.g.* silent movement, off-camera speech, ...).

Chapter 5

# Additional contributions

INTRODUCTION DE CE CHAPITRE A FAIRE

Second, we present additional anomaly detection results for models still in an early development stage, yet promising. Third, we discuss representation learning, what it is and how essential it will be for anomalous human behavior detection in the future. Finally, we discuss the efforts we have made to ensure the reproducibility of all the work presented in this thesis and draw a parallel with previous state-of-the-art work which, for the most part, does not ensure this.

## 5.1 Additional Prospective Models and Discussion

In this subsection, we present additional models and results that were developed during this thesis. While these methods are not as advanced and developed as the previous ones, these additional methods address some aspects that were left over by the main methods and should be covered as a follow up to this thesis.

First, we present methods directly aiming at jointly modeling audio and video in order to detect anomalous behaviors. Then, in the objective of improving the adaptability of a method to problems such as concept drift, active learning and new contexts, we present some meta-learning methods. Finally, we discuss the notion of priors in models and how they probably impact models for anomaly detection.

### 5.1.1 Audio-video models

The IAE and the VIAE were specifically developed to model the temporal aspect of human behaviors. The CnC was built with a focus on the general definition of anomalies. So far, we have not described any method specifically developed for joint multimodal detection of anomalous human behaviors.

We present three methods we built with this goal in mind. To our knowledge, they are

the first methods to be specifically developed to that end. For each method, we follow a similar pattern as before to describe them: First, we start with the motivation behind the method. Second, we describe how a model is built and trained using the method. Third, we describe how anomalies are detected using the model and, when they exist, we present the results on the Emo&ly dataset. Finally, we talk about future tests and improvements that could be carried out to assess the method.

#### 5.1.1.1 Domain adaptation for cross-modal generation

The first method we developed for audiovisual data aims at performing domain adaptation where each modality is considered to be a domain.

**Motivation.** When they are aligned, the audio and video tracks share a significant amount of mutual information. The goal of this method is double: model the mutual information between audio and video, and to propose a way to detect anomalies. We assume we can model this mutual information by learning to infer or generate one modality based on the other.

**Architecture and training.** For this method, we mainly draw inspiration from M.-Y. Liu, Breuel, and Kautz 2017 who worked on image-to-image translation. Image-to-image translation is an application of domain adaptation, where the two sets of images come from different domains. In our case, we consider the two different domains to be audio and video and aim to translate between these two domains.

The method introduced by M.-Y. Liu, Breuel, and Kautz 2017 is made of two VAE-GANs who share a common latent space. We presented VAE-GANs earlier in Section 3.2.2. The sharing of the latent space allows to connect the output of any of the two encoders to the input of any of the two generators, as illustrated in Figure 5.1. Additionally, thanks to the presence of discriminators, we do not require pairs of aligned tracks. The video and the audio tracks do not need to be aligned or come from the same source.

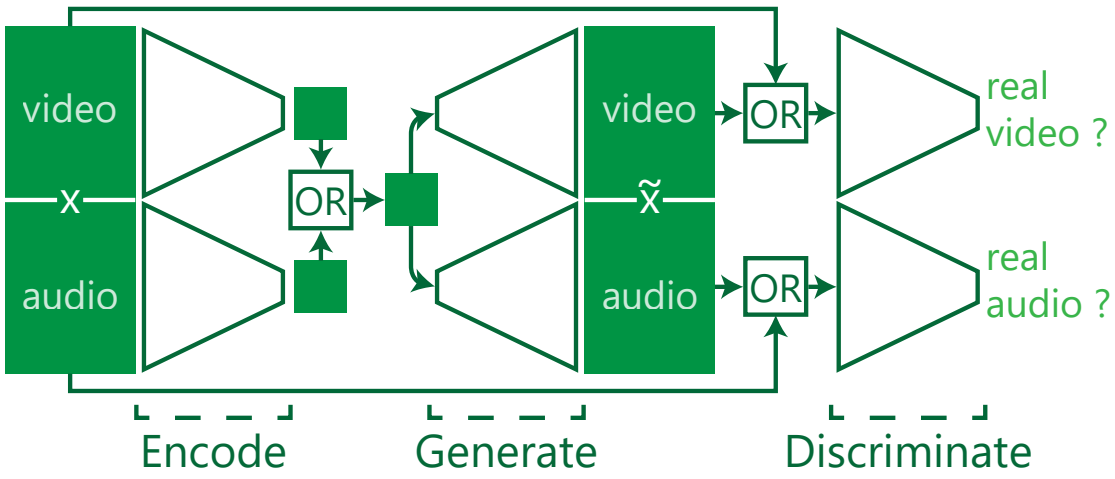


Figure 5.1 – In VAE-GANs, both domains — audio and video here — share the same latent space. Once a modality has been encoded, it can be decoded by either of the two decoders. The discriminators provide feedback to the encoders and generators, enabling us to train this model.

The main difference between our work and the work of M.-Y. Liu, Breuel, and Kautz 2017 is the usage of different modalities. This implies the encoders, generators and discriminators have different architectures and complexity. Moreover, the shared latent space is more constrained as audio and video have to share the same dimensionality for their latent codes, even if one may contain more information than the other. Having two different modalities also creates an unbalance in the loss terms during training. For example, the norm of the reconstruction losses will significantly differ from audio to video. This means, if left unchanged, one modality can have significantly more weight than the other one when training the model. There exist several methods to correct this balance problem, such as dynamically weighting the reconstruction errors by normalizing them, however their impact would have to be studied.

**Anomaly detection.** Since this method uses two VAE-GANs, we have several options to compute anomaly scores:

1. We can use the discriminators as-is on the incoming data. The sum of the outputs of their outputs will yield the anomaly score: the more the data looks fake to them, the greater the anomaly score.

2. Similarly, the VAEs can be used as autoencoders on the incoming data. The sum of the reconstruction errors of both autoencoders would yield the anomaly score.
3. Instead of directly using the incoming data for the previous options, both modalities are first adapted to the domain of the other modality.
4. Instead of using the scores yielded by the discriminators and the VAEs separately, they can be merged.

**Future works.** As with most — if not all — adversarial models, instability can occur during training when either the discriminator or the generator starts beating the other one. It will be desirable to investigate how stability can be improved with other formulations of GANs, such as the Wasserstein GAN (W-GAN) (Arjovsky, Chintala, and Léon Bottou 2017) or the Wasserstein GAN with Gradient Penalty (WGAN-GP) (Gulrajani et al. 2017).

Additionally, while we provided several options to compute anomaly scores, future works should investigate which options are better suited for anomaly detection and their relative performance with other models.

#### 5.1.1.2 Results

Since we did not reach a point where the method is mature enough to perform anomaly detection, we do not have anomaly detection results to present for this method.

#### 5.1.1.3 Energy-based models for cross-modal consistency

The second method we developed for audiovisual data aims at modeling behaviors by estimating the consistency between the audio and video tracks.

**Motivation.** We formulate a new method inspired by Energy-based GANs (EBGANs) (J. Zhao, Mathieu, and LeCun 2016). While the EBGAN is a monomodal method, the method we present here is a multimodal method. Contrarily to GANs, EBGANs can use any energy function as their discriminator, including the reconstruction error of an autoencoder. The discriminators in EBGANs are trained to reconstruct all inputs, until a given threshold. After that, their goal is to keep improving on compatible data and keep the reconstruction error at the threshold for uncompatible data.

The previous method models both audio and video in a single latent space, regardless of the relative difference between the amounts of information they contain. This is due to the architecture used in this method. As we will detail later, in this method audio and video can have latent spaces with different sizes and still be modeled jointly.

**Architecture and training.** Since our method is based on EBGANs, we have to build an autoencoder that will be used for the energy-based discriminator. To that end, we build a multimodal autoencoder in three parts, similarly as we proposed for an eventual multimodal IAE and illustrated in Figure 4.11.

The first part is made of two encoders $e_{audio}$ and $e_{video}$, one for each modality. The second part is a deeper autoencoder made of the encoder $e_{joint}$ and the decoder $d_{joint}$. The third and last part of the energy-based discriminator is made of two decoders $d_{audio}$ and $d_{video}$.

However, instead of training the energy-based discriminator to separate real from generated samples, we are going to train it to do what energy-based discriminator do best: estimate the compatibility between inputs. We are only going to give the discriminator one generated modality at most. Therefore, the task of the adversarial generator becomes to generate a modality that is compatible with a real modality. For example, the generator can be tasked to generate an audio track that is compatible with a pre-existing video track.

In order to do that, we modify the way the adversarial generator is set up, yielding a generator that is more inspired by VAEs and VAE-GANs than by EBGANs. This means we add an encoder before the generator, this way the generator becomes conditioned on latent codes resulting from real data. However, instead of creating one additional encoder for each modality, we re-use the encoders of the energy-based discriminator ($e_{audio}$ and $e_{video}$). On the other hand, we create one generator for each modality, namely $g_{audio}$ and $g_{video}$.

When encoding $x = (x_{audio}, x_{video})$, we get the latent codes $z_{audio} = e_{audio}(x_{audio})$ and $z_{video} = e_{video}(x_{video})$. If we consider $z_{joint} = (z_{audio}, z_{video})$, by autoencoding $z_{joint}$, we get $\tilde{z}_{joint} = (\tilde{z}_{audio}, \tilde{z}_{video}) = d_{joint}(e_{joint}(z_{joint}))$.

We can then decode $\tilde{z}_{audio}$ and $\tilde{z}_{video}$, yielding $\tilde{x}_{audio} = d_{audio}(\tilde{z}_{audio})$ and $\tilde{x}_{video} = d_{video}(\tilde{z}_{video})$.

On their side, the generators $g_{audio}$ and $g_{video}$ receive the latent codes $z_{audio}$ and $z_{video}$, then produce the generated outputs $\hat{x}_{audio} = g_{audio}(z_{audio})$ and $\hat{x}_{video} = g_{video}(z_{video})$.

All these operations are illustrated in Figure 5.2.

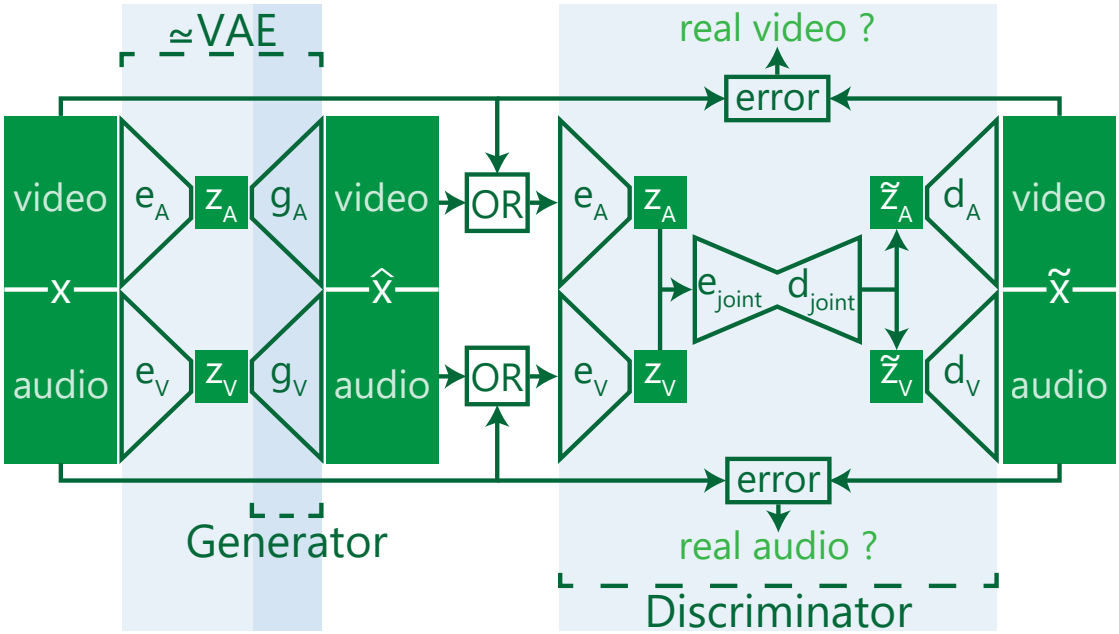


Figure 5.2 – On the left, the generators aim to produce realistic samples based on latent codes given the discriminator's encoders. On the right, the discriminator aims to separate real samples from generated samples by yielding higher reconstruction errors for generated samples than for real samples. The presence of a joint model within the discriminator ensures consistency between the two modalities.

During training, one of the two input modality — $x_{audio}$ and $x_{video}$ — is switched with their generated counterparts — $\hat{x}_{audio}$ and $\hat{x}_{video}$ — to produce the incompatible inputs for the energy-based discriminator. Additionally, there are as many compatible inputs than their are incompatible inputs to preserve balance.

Assuming a given input modality $x_m \in \{x_{audio}, x_{video}\}$, the reconstruction error for this modality is given by:

$$\mathcal{E}_m(x) = ||x_m - \tilde{x}_m||^2 \tag{5.1}$$

Following Equation 5.1, the energy function is defined based on the individual reconstruction error for each modality as follows:

$$\mathcal{E}(x) = \mathcal{E}_{audio}(x) + \mathcal{E}_{video}(x) \tag{5.2}$$

Using $(a, b)$ to represent tuples and given Equation 5.2, the generator is trained to minimize $\mathcal{L}_{gen}$:

$$\mathcal{L}_{gen} = \frac{\mathcal{E}((\hat{x}_{audio}, x_{video})) + \mathcal{E}((x_{audio}, \hat{x}_{video}))}{2} \tag{5.3}$$

Given Equations 5.2 and 5.3, the energy-based discriminator is trained to minimize the following adversarial loss $\mathcal{L}_{disc}$:

$$\mathcal{L}_{disc} = \mathcal{E}(x) - \mathcal{L}_{gen} \tag{5.4}$$

It is worth noting, since the encoders are part of the energy-based discriminator, we do not want to train them to fool the discriminator. Therefore, the encoders are set up to only receive updates from $\mathcal{L}_{disc}$.

**Anomaly detection.** With this method, anomaly scores could be obtained from the energy function $\mathcal{E}$ modeled by the energy-based discriminator. Anomalies should obtain higher energy values than normal samples.

However, the energy-based discriminator itself may only be able to detect if the tracks match, which should also be the case for anomalous behaviors. Therefore, we propose to use the generators first and use $\mathcal{L}_{gen}$ as the anomaly score. Having learnt to generate sample conditioned on real samples, the generators should produce less compatible samples for anomalies, which would then be reflected in $\mathcal{L}_{gen}$.

Unfortunately, we did not manage to reach the anomaly detection phase with this method. In our experiments, the generators suffered from what is called *mode collapse*, a situation where the generators always seem to produce the same output. As with the previous method, the adversarial component often leads to an unstable model, which requires a careful setup to overcome.

**Future works.** Besides the improvement to stability and further experiments to asses the validity of the method, there is a simple yet important change to investigate in future works. Currently, the generators are conditioned on the same modality they produce. The joint modeling would likely be improved by doing the opposite: generating video from audio and vice-versa.

As the latent spaces of the different modalities have different sizes, we would have to constrain the model by sharing the latent spaces. Otherwise, we could also train a new set of encoders to perform this cross-modal projection. However, as-is, both cases are likely

to increase the instability of the model further.

#### 5.1.1.4 Transformers for cross-modal generation

The third and last method we developed for audiovisual data is a variant of the multimodal IAE we drew in Section 4.2.9 where a transformer is added to perform both modality prediction and domain adaptation.

**Motivation.** Similarly to the first audiovisual method we presented, we assume domain adaptation to be a valid option to improve the joint modeling of multiple modalities. Additionally, as Y. Zhao et al. 2017 showed with their STAE — a method we presented in Section 3.2.1.1 —, training a model to predict the future can improve its ability to model normality in relevant contexts.

Transformers (Vaswani et al. 2017) were originally designed for translation. The encoder of the transformer takes as input the source language while the decoder both receives and outputs the target language. The transformer then produces each word of the target language one by one. Each time a word is produced, it is added to the input of the decoder, conditioning the remaining of the translation.

Our method is based on this design and changes two things for our purpose: First, it replaces the source language by a source modality and the target language by a target modality. Second, its starts the translation midway, meaning the decoder already has access to the first half of the target modality and it has to predict the other half.

In other words, the problem we give to the model is: "Given one full audio track — respectively video track — and one partial video track — respectively audio track —, fill in the blanks of the partial track."

**Architecture and training.** For building the model, we first train a multimodal IAE in order to facilitate the training of the transformer. Preferably, this pre-trained model contains a fusion sub-model responsible for modeling the joint latent codes produced by the encoders of each modality.

Once the multimodal IAE is pre-trained, each modality $x_m, m \in \{audio, video\}$ is encoded into their respective latent codes $z_m$. We then build one encoder for our transformer for one modality and one decoder for the other modality.

Decoders in transformers are autoregressive models. This implies they yield their outputs one by one. At each iteration, the decoder first looks at the outputs it already knows

— or has predicted —, then looks at the output of the encoder and finally produces the next output element based on these two information.

As illustrated in Figure 5.3, since — in our case — the transformer is trained to only predict the second half of one track, its decoder starts with the elements of the first half of the same track already filled in.

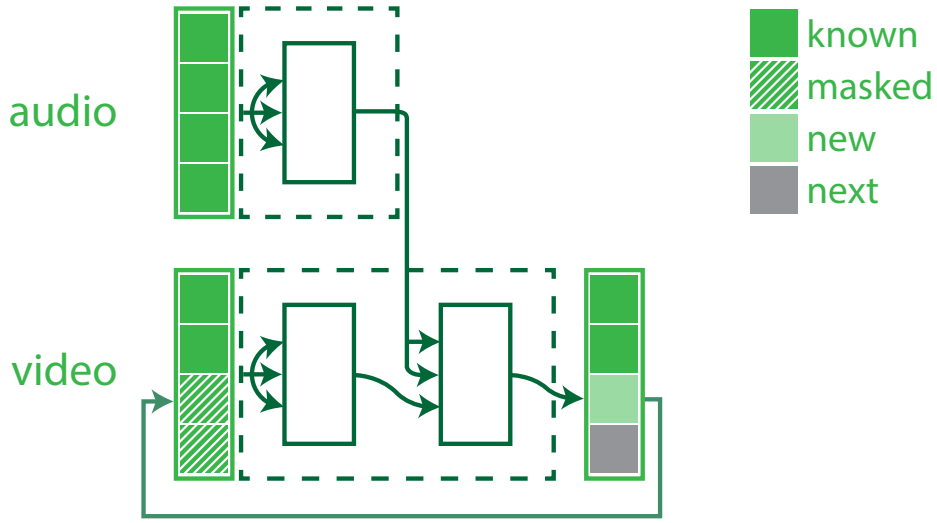


Figure 5.3 – In this example, the second half of the video track starts masked. The transformer first looks at the audio and video tracks independently. It then combines them to produce the next element in the video track. Finally, this new element is added to the known video track and the cycle repeats until the video track is full.

In a similar way as previous models, here the loss function minimized by the transformer is based on the reconstruction error, or rather prediction error.

We note that, in this setup, the transformer is only able to adapt domains in one direction as both parts of the transformer can only handle one modality.

**Anomaly detection.** In the current configuration of this method, the only metric we really have access to is the prediction error. Therefore, when using this method, anomaly scores are computed from the errors made by the model when predicting the future of a given modality.

However, it may be possible to improve anomaly scores by having two transformers, one for each direction of the domain adaptation. In our case, this would mean one model for "audio to video" and one model for "video to audio". In this configuration, anomaly scores would be computed from the sum of prediction errors for each modality.

Additionally, a balancing factor should be introduced as, in our experiments, the norm of errors of audio and video widely differed. To do so, we would compute normalization factors for the prediction errors on audio and video separately on the training set. Then, we would use these normalization factors to compute the balancing factor for the testing set and subsequent applications.

**Future works.** As with the previous methods presented in this section, more work is necessary on the training the model and to assess the overall validity of the method.

Additionally, future works could investigate the extension of this method to two-way domain adaptation. For example, we could discard the encoder and have a bimodal decoder where the modality to predict is masked but not the source modality. This way, the direction of the domain adaptation can be changed by simply masking the other modality. We illustrate this alternative in Figure 5.4.

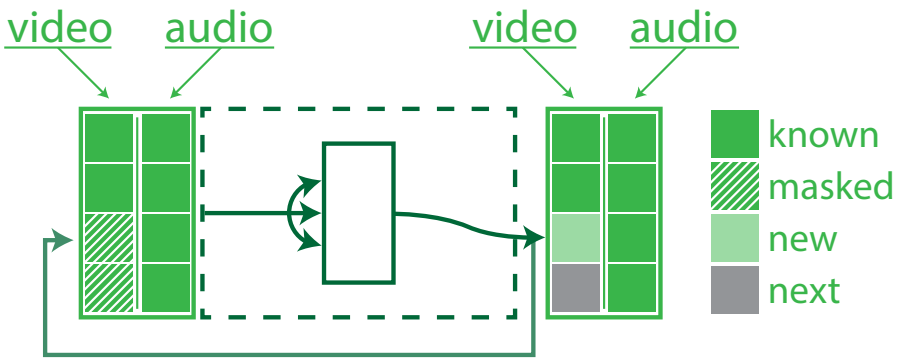


Figure 5.4 – In this setup, only the decoder of the transformer is kept. Here, we only represent one stage of said decoder. In this example, half the video track started masks. The transformer produces the next element of both tracks but only the element of the masked track is kept and added to the next input, as the other track is already fully known.

### 5.1.2 Meta-learning models

In Section 3.2.5, we talked about meta-learning models. As we stated, they are notably interesting to anomaly detection for their ability to adapt quickly. This ability is helpful against concept drift, useful for deploying the same model in new contexts and potentially improve the training of other models.

We adapted two meta-learning models for anomaly detection: Memory-Augmented Neural Networks (MANNs) and Kanerva machines. We detail both adaptations and their resulting models in four parts: 1) The motivation behind, 2) an overview of the architecture, 3) how the model can be used for anomaly detection and 4) the results we were able to achieve.

#### 5.1.2.1 Adapting MANNs for anomaly detection

MANNs are meta-learning models originally designed for training classifiers based on only a few samples from each class (Santoro et al. 2016). This is commonly referred to as few-shot learning. We presented the original work in more detail in Section 3.2.5. Here, we present our adaptation of this work for anomaly detection.

**Motivation.** While MANNs are meta-learning models, they are not the only type of meta-learning models. What differentiate MANNs from other models is the presence of an external memory. This external memory sort-of acts like RAM in a computer, in the sense it is independent of the software using it. For a deep learning model, this means the model has now additional parameters that are not tuned during training but by inference. In practice, this is made by writing during inference to an external memory that contains these parameters, in a differentiable manner. Where the tuning of parameters through gradient descent during training is slow, the tuning of the parameters in the external memory can be fast and more efficient.

This observation gives us a new — more flexible — way to learn normal patterns and store them in an external module. This gives us two potential improvements over other methods. First, not related to anomaly, is the possibility to just replace the external memory when facing a new context of application, instead of retraining the model entirely. Second, as we only want to store normal patterns in the external memory, we expect this can be helpful for anomaly detection, as anomalies should not have matching patterns in the external memory.

However, MANNs are originally intended for classification. Therefore, we modify them to turn them into autoencoders. We chose to turn them into autoencoders because it is a well documented and competitive framework for anomaly detection.

**Architecture.** MANNs are already equipped with an encoder, so we just need to replace the last part of the model by a decoder. Then, we can encode the input, refine the latent codes based on the memory, update the memory and decode the refined latent codes

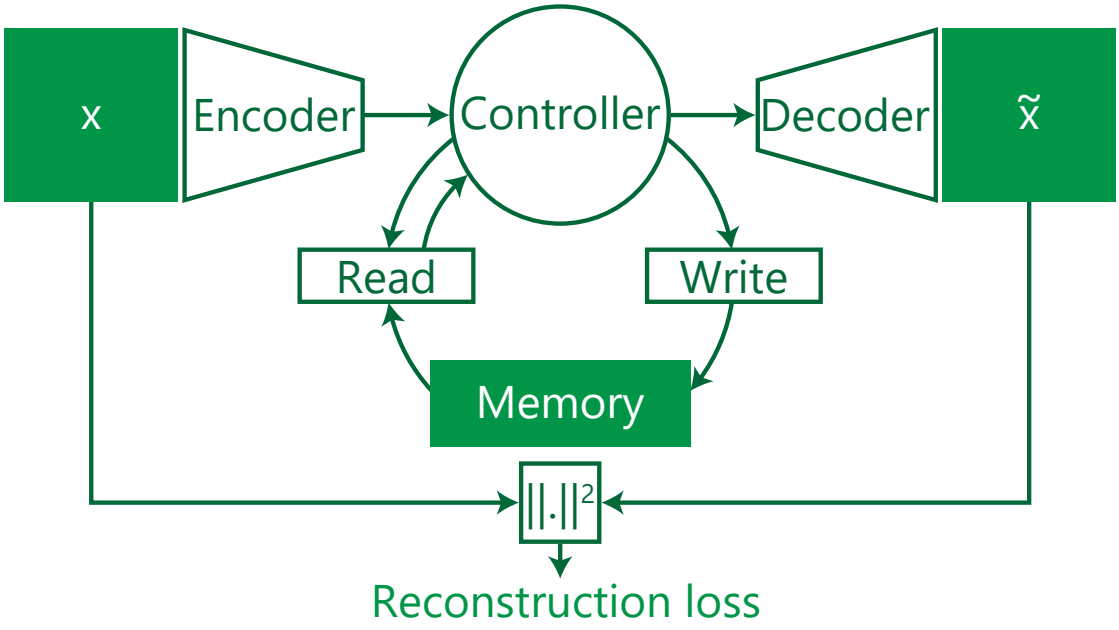


Figure 5.5 – We adapt a MANN by including a decoder and replacing the loss term by a reconstruction loss term.

It is worth noting that, during training, the external memory is wiped at every train step. For each train step, the model performs several autoencoding iterations and is trained to reconstruct all samples it is presented with.

As the memory starts empty, reconstructing samples for earlier iterations is harder than for later iterations. Therefore, we add an increasing weight to the reconstruction loss of individual samples, where this weight increases after each iteration.

**Anomaly detection.** Once the model has been trained, we stop wiping the memory. Instead, the model first performs inference on a given amount of normal data, filling

the memory (partially or fully). Then, the memory is frozen and it is used as a regular autoencoder where the reconstruction error gives the anomaly score.

For deployment to production, the memory is unfrozen. However, for each sample, it is only updated when the overall system does not flag the sample as an anomaly, hopefully keeping anomalous patterns out of the external memory.

**Results.** We trained this model on the UCSD Pedestrian 2 dataset to serve as a proof of concept. In Figure 5.6, we report the reconstruction error ratio between normal samples and anomalies over the training phase. The resulting curve indicates the model learns to better reconstruct normal samples than anomalies, despite its external memory being wiped after each train step. This suggests this model could be used to detect anomalies.

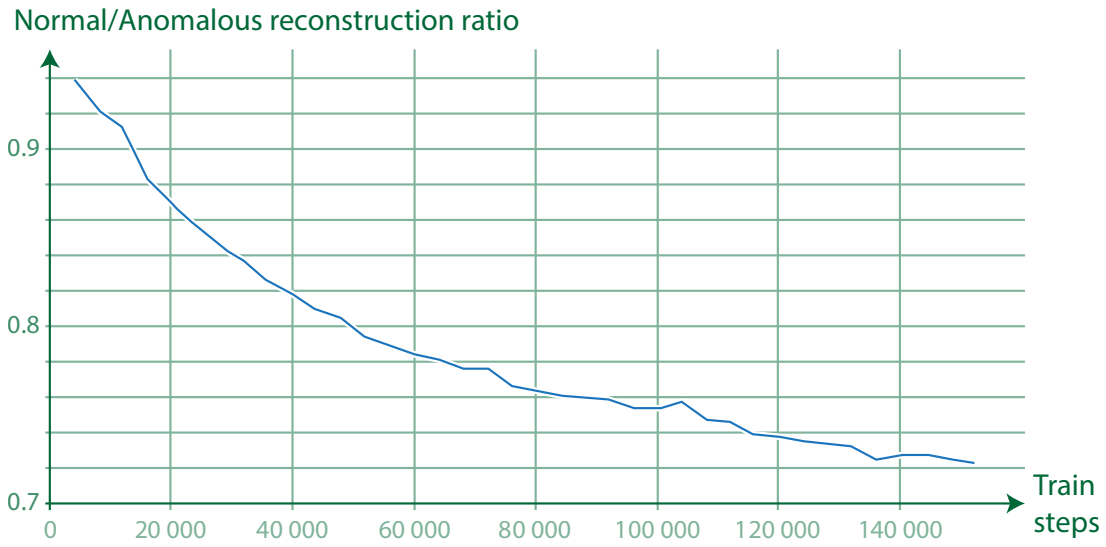


Figure 5.6 – We plot the reconstruction error ratio between normal samples and anomalies as the model is iteratively trained. As the model is trained, the reconstruction of normal samples improves faster than of anomalies.

However, this is a meta-learning model and we only trained it on a single dataset with a single scene. It is possible this model learns to function as a classical autoencoder, storing the context in its internal weights instead of the external memory. Therefore, further investigation is required to better understand MANNs in the context of anomaly detection.

#### 5.1.2.2 Adapting the Kanerva Machine for anomaly detection

Kanerva Machines (Y. Wu, Wayne, Graves, et al. 2018) are MANNs inspired by Kanerva's sparse distributed memory. However, contrarily to previous MANNs, Kanerva Machines are conditional generative models. Put simply, they are memory-augmented VAEs.

**Motivation.** Our motivation for investigating the Kanerva Machine for anomaly detection is twofold:

First, contrarily to previous MANNs, Kanerva Machines have autoencoding capabilities by default which we can use to compute a reconstruction error and therefore an anomaly score.

Second, the Kanerva Machine allows for iterative sampling by feeding back the model's output — the reconstruction — as input. According to the authors of the Kanerva Machine, while not proven, doing so should converge to a pattern stored in memory. For anomaly detection, this may be a way to access known normal patterns and differentiate them from anomalous patterns.

**Architecture.** At the moment of writing, we do not plan any significant modification when adapting the Kanerva for anomaly detection. However, since the Kanerva Machine is originally meant for images, we adapt the encoder and the decoder for the target modality (*e.g.* audio, video...). In Figure 5.7, we show the different graphical models illustrated in the original paper, with slight modifications to better match our explanation.

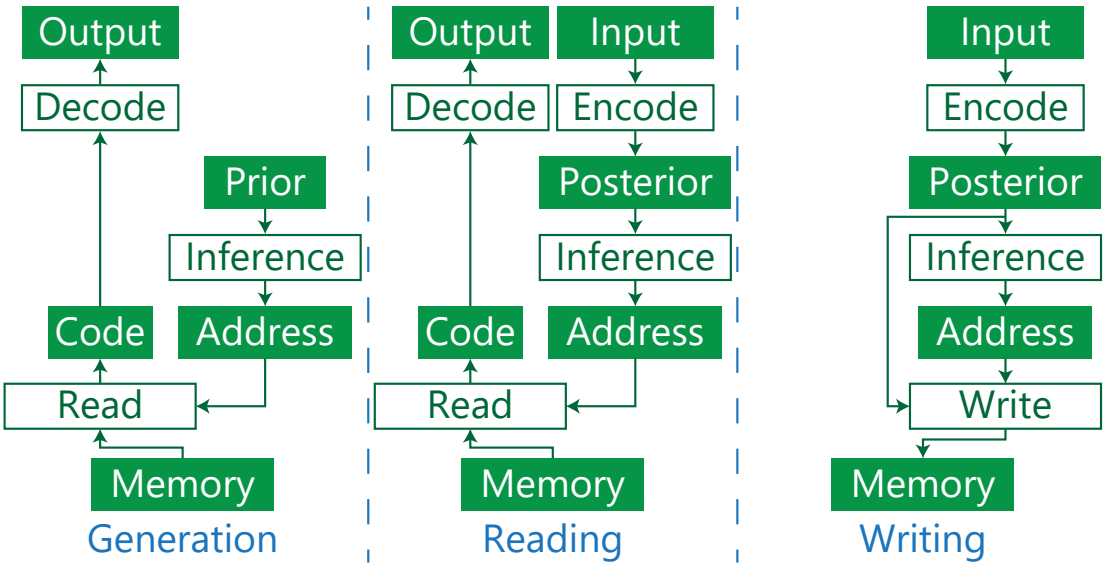


Figure 5.7 – Views of the three mains processes in the Kanerva Machine as presented by the authors. The central process — reading — can be used as an autoencoder. Iteratively using this process is expected to converge to a stored pattern in memory.

It is worth noting that the images used by the authors of the Kanerva Machine for experimenting on it where rather small (32x32 at most). We think this input size is unlikely to be enough for detecting anomalous behaviors in other modalities such as video or audio. Because of this, we plan on pre-training an autoencoder, such as an IAE, in order to first reduce the dimensionality of the input data, therefore making it more manageable for the Kanerva Machine.

**Anomaly detection.** Once the training is complete, the model is fed with normal samples in order to fill the memory in a similar way as the one we explained for MANNs. For the Kanerva Machine, this means repeatedly using the right process — writing, illustrated in Figure 5.7 — on a set of normal samples.

Again, similarly to what we described for MANNs, we can use the Kanerva Machine as an autoencoder to produce anomaly scores from reconstruction errors.

However, the iterative sampling capabilities of the Kanerva Machine give us a new potential way to detect anomalies. Any sample — normal or anomalous — iteratively sampled should be converted into a normal sample. We expect this would create a large reconstruction error between the result and the original sample if it is anomalous.

Additionally, we could count the number of sampling iterations required for a pattern to converge — until a maximum number of iterations if it does not converge — or the total distance covered by the sampled patterns.

**Results.** Because we did have access to the source code of the Kanerva Machine, and because the architecture is rather complex, we had difficulties to implement it. Once implemented, we did not manage to get a model to converge during training on small videos.

At the moment, we only scratched the surface of what is possible to do with the Kanerva Machine — and other memory-augmented methods — for anomaly detection. There are still many leads to investigate on this subject.

### 5.1.3 Model priors

In this thesis, we are interested in multiple modalities. A general solution — that can be adapted with little to no effort to any modality — is very appealing, even in the cases where we do not jointly model multiple modalities.

Therefore, since CNNs can be easily adapted to any modality, we started under the assumption that they are a general solution. But more importantly, we assumed CNNs have no priors on either normal or anomalous samples, and that only the context fed to the model during training would matter in this regard.

Ulyanov, Vedaldi, and Lempitsky 2018 showed that, contrary to the popular belief that the performance of CNNs is solely due to their ability to learn good priors, CNNs have strong structural priors.

#### 5.1.3.1 Motivation

We first want to replicate the work of Ulyanov, Vedaldi, and Lempitsky 2018, in order to validate their findings and gain insights of what it could mean in our case since we study multiple modalities.

#### 5.1.3.2 Method

The method introduced by Ulyanov, Vedaldi, and Lempitsky 2018 is rather straightforward. First, a model $f_\theta$ made of CNNs is built and is parameterized by the weights $\theta$. This model is fed with a noise tensor $z$, yielding the output $f_\theta(z)$. Then, a differentiable

corruption function is defined, we call this function $c$. $c$ can be any differentiable function, such as down-sampling, noising, masking, *etc.* Finally, both the input image $x$ and the model's output $f_\theta(z)$ are corrupted by $c$. The goal of the model is to minimize the difference between $c(x)$ and $c(f_\theta(z))$. We illustrate one step of this process in Figure 5.8. This step is repeated until convergence or a given number of iterations.

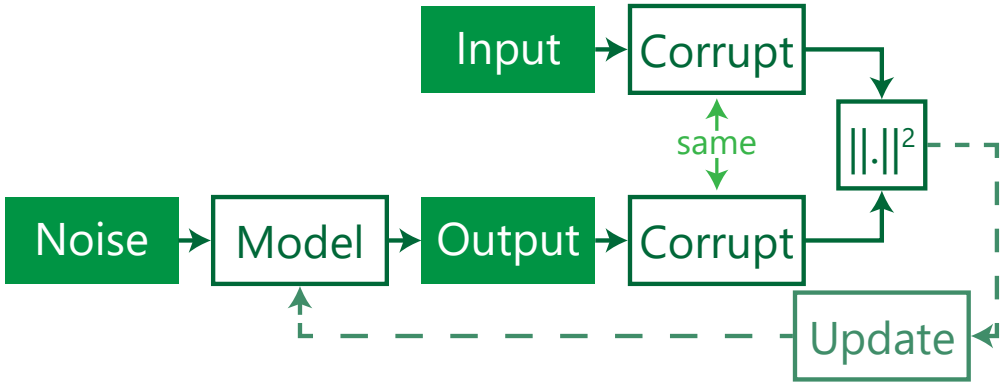


Figure 5.8 – The model's parameters are fit to minimize the difference between its output and the input image after they have been corrupted by the same process.

It is important to note that, contrarily to usual deep learning training phases, the model is not fit on a dataset but on a single sample. Therefore, the model has not learnt any prior before, leaving room only for structural priors.

#### 5.1.3.3 Results and future works

First, we were able to easily replicate the work of Ulyanov, Vedaldi, and Lempitsky 2018. For example, we masked parts of an image and the method was able to fill in the blanks in a convincing way.

Second, this invites us both to be more careful when choosing an architecture but also to look for architecture that may have positive biases for our task. In particular, the overall structure of the IAE and the VIAE is meant to induce better priors for sequential data. Since human behaviors are mostly observed as sequential data, this findings may partially explain the success the IAE and the VIAE.

Additionally, since such prior will condition the latent representation learnt during training, this gives us a better insight about representation learning for anomaly detection, which we will cover in the next part.

## 5.2 Representation Learning

We now bring the subject of representation learning. We believe it is necessary to explore this topic in order to reach satisfying anomalous human behavior detection.

In this part, we first bring up the problems posed by measuring distances in the input space as it is most commonly done in the literature and how representation learning can help with these problems.

Second, we discuss the abstract aspect of the notion of behavior itself and how representation learning can help better identify behaviors and therefore separate normal behaviors from anomalous behaviors.

Finally, we present a few representation learning models that exist in the literature and that can be of use for modeling representations of human behaviors.

### 5.2.1 Distance measures in the input space

As we have seen before, in the different stages of anomaly detection we often need the ability to compare two samples to measure their similarity. For example, when using autoencoders for anomaly detection, we compute a distance between the inputs of the autoencoders and their corresponding outputs. This is done during both training and during the detection phase. During training, the goal of the autoencoder is to learn to minimize this distance. During the anomaly detection phase, this distance is used as the anomaly score to determine whether a sample is normal or anomalous.

#### 5.2.1.1 Element-wise error measures in the input space

The most common distance measures found in the literature, for either training an autoencoder or computing anomaly scores, operate element-wise (e.g. pixel-wise for images and videos). The $\ell_1$ and the $\ell_2$ norms are by far the most common (Sakurada and Yairi 2014; Marchi et al. 2015; An and Cho 2015; Hasan et al. 2016; Sabokrou, Fathy, and Hoseini 2016; D. Xu et al. 2017; Chong and Tay 2017; W. Liu et al. 2018). The $\ell_1$ norm of the error is commonly referred to as the Mean Absolute Error (MAE) and the $\ell_2$ norm of the error as the Mean Squared Error (MSE). We also observe the element-wise cross-entropy error can be used on rarer occasions (Yousefi-Azar et al. 2017).

Many reasons why the MSE is so popular are explained by H. Zhao et al. 2016, these reasons include ease of use, well-known properties and generally good results. Element-wise error measures also have the benefit of allowing the localization of the source of the

error (e.g. the reconstruction error of an image can indicate the location of an anomaly on the image).

#### 5.2.1.2 Other metrics in the input space

However, element-wise error measures also have some significant drawbacks.
For example, it is well known that MAE, MSE and PSNR (which derives from MSE) poorly correlate with human's perception of visual quality (Z. Wang, Bovik, et al. 2004; L. Zhang et al. 2012). Motivated by this observation, Z. Wang, Bovik, et al. 2004 developed the Structural Similarity Index Measure (SSIM) for grayscale images, measuring luminance, contrast and structural distortions separately.

Variants of the SSIM were later developed to overcome a few of its shortcomings: Z. Wang, L. Lu, and Bovik 2004 developed the Video-SSIM, a generalization of the SSIM to video. The Feature Similarity Index Measure (FSIM) (L. Zhang et al. 2011), which exploits the dissimilarities in phase congruency and gradient magnitude, has been extended to $\text{FSIM}_c$ in order to be used on colored images.

Unfortunately, to our knowledge, none of the variants of SSIM are differentiable, except for the original SSIM itself. Differentiability is a requirement for training neural networks through gradient descent. It is worth noting that differentiable measures could be first used for training the model and non differentiable measures afterward for computing anomaly scores. Our results with the IAE, which was trained using MSE, show that SSIM is sometimes preferable over using MAE and MSE for computing anomaly scores.

However, SSIM and its variants have limited interest in our case for two reasons: 1) They cannot be applied to other modalities besides images and videos and 2) they are meant for computing visual similarity, not behavioral similarity.

#### 5.2.1.3 A simple failure case

In Figure 5.9, we present a simple case with three images from the Emo&ly dataset. In this case, a different (and anomalous) behavior from another scene is considered more similar to a normal behavior than the same normal behavior where the edges of the image were cropped and replaced with pure black. Values for MAE, MSE, PSNR and SSIM can be found in Table 5.1.

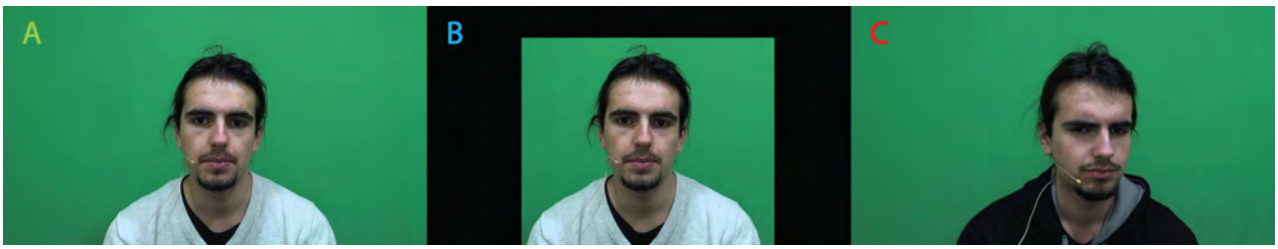


Figure 5.9 – A) The reference image, the subject displays a normal behavior. B) Cropped version of the reference image, the subject still displays a normal behavior. C) The subject is consciously acting an anomalous behavior. All images were taken from the Emo&ly dataset. Images A and B come from the Segment 5 of the "Normal" round of subject n°25. Image C comes from the Segment 10 of the "Acted" round.

| | 1) Normal vs Cropped | 2) Normal vs Anomalous |
|---|---|---|
| MAE | 0.18 | **0.10** |
| MSE | 0.08 | **0.05** |
| PSNR | 11.0 | **13.1** |
| SSIM | 0.52 | **0.81** |

Table 5.1 – Values given by MAE, MSE, PSNR and SSIM when measuring similarity between the reference image, displaying a normal behavior and 1) a cropped version of this image, and 2) an image displaying a different (and anomalous) behavior. For MAE and MSE, lower is better. For PSNR and SSIM, higher is better. The best similarity values are in bold.

We note that, even if we are mainly interested in anomalous behavior detection here, the example presented in Figure 5.9 and Table 5.1 is also relevant for general measures of visual similarity. Not only the behavior from image C is different from the behavior from images A and B, but C is also more dissimilar visually from A than B is.

#### 5.2.1.4 A space of representations for behaviors

One of the reasons for using representation learning is to overcome those shortcomings. Properly conditioned, representation learning should be able to model behaviors in their own latent space, allowing projection from the input space to a "behavior space" and the usage of measures like $\ell_1$ and $\ell_2$ in this latent space.

### 5.2.2 The notion of behavior is abstract

What separates a human behavior from another one ? How can we draw such a line ? How do we automate this process ?

These are essential questions to answer in our case, because if we cannot characterize behaviors, how can we expect to classify some behaviors as normal and others as anomalous ?

As pointed out by Skinner 1965, we may all be very familiar with the notion of human behavior as we have observed it for many years, yet *we are not necessarily able, without help, to express useful uniformities or lawful relations. [...] Behavior is a difficult subject matter, not because it is inaccessible, but because it is extremely complex. Since it is a process, rather than a thing, it cannot easily be held still for observation.*

We may be able to have a case-by-case intuition about what human behavior we are currently observing. We may be able to label this behavior we are observing, and even give some reasons why we chose this label but this is only intuition. However, the difficulty is to move away from this intuition and *express useful uniformities*. The same goes for deciding if a behavior is anomalous or not: It may be easy to do it intuitively case-by-base but this is not helpful in automating the detection of anomalous human behaviors.

However, as we will see, a model can sort-of learn these intuitions using representation learning. In this case, intuitions would take the form of latent representations. This would provide us a way to automate our task. Additionally, as we will see afterwards, this is also an opportunity to gain insight about the model's decision and therefore allow for some interpretability of the decision.

#### 5.2.2.1 Self-supervised modeling of behaviors

To our knowledge, self-supervised learning is currently the best solution for achieving good representation learning. By "good", we mean different views of the same input should robustly yield close representations. For example, the camera angle should not impact the representation of a behavior as long as the behavior can be properly observed.

As we discussed in Section 3.1.4, self-supervised learning consists in exploiting the order present in the data. This is done by learning representations that are robust to unrelated changes, such as using a different view of the same object.

A well trained representation model of human behavior using self-supervised learning should map the same behavior to the same point in the latent space, no matter the noise

intrinsically present in the context (*e.g.* captor noise, weather, lighting, ...) Similar behaviors should find themselves closely mapped to each others in the latent space, therefore allowing us to compare them. While not drawing a line so to speak, comparisons in such a space could be enough for performing satisfying anomalous behavior detection.

#### 5.2.2.2 Improve interpretability for better anomaly detection systems

Additionally, self-supervised learning could improve the interpretability of the decision of an anomaly detector. This would especially be the case in a multimodal setup where each modality can be seen as a different view of the same behavior.

In Figure 5.10, we illustrate an example where all modalities of a single behavior are mapped to a single latent representation, similarly as to what we saw for domain adaptation using VAE-GANs. Then, the representation of a given anomaly is compared to the representation of known normal samples, yielding a new representation that may help understand why this sample is considered anomalous.

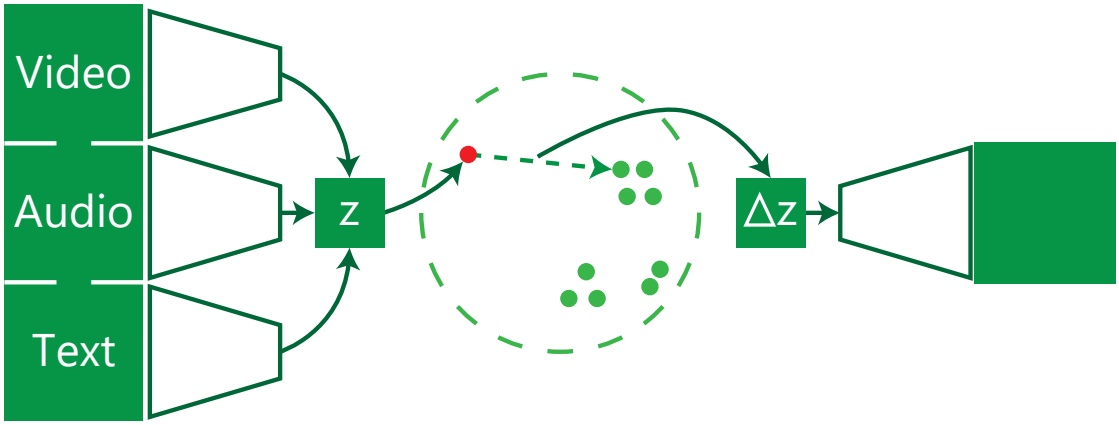


Figure 5.10 – The representation of an anomalous sample is compared to the representation of normal samples. This comparison may yield insights on the model's decision.

Such difference is a vector and could be interpreted in the same way it is possible with the relationship vectors of the popular method word2vec (Mikolov et al. 2013), where relationship vectors also had meaning (*e.g.* the woman-man relationship vector: $f(sister) - f(brother) = f(granddaughter) - f(grandson)$).

Moreover, representations can be used to find samples with similar representations. In the case of anomalies, this could result in yielding the representations of previously

observed anomalies — if they were stored — or in yielding no representation, effectively saying: "no similar behavior — even anomalous — were ever observed before".

### 5.2.3 Representation models

We now present four methods from the literature for representation learning. While the first one does not use self-supervised learning, the three other do. Additionally, the last method we present here is for multimodal representation learning.

#### 5.2.3.1 Autoencoding beyond pixels

We already talked about this work previously as this is the work that introduced VAE-GANs. We illustrate in Figure 5.11 the graphical model of VAE-GANs we showed in Section 3.2.2, with an emphasis on the perceptual loss and the latent representation obtained from the discriminator.

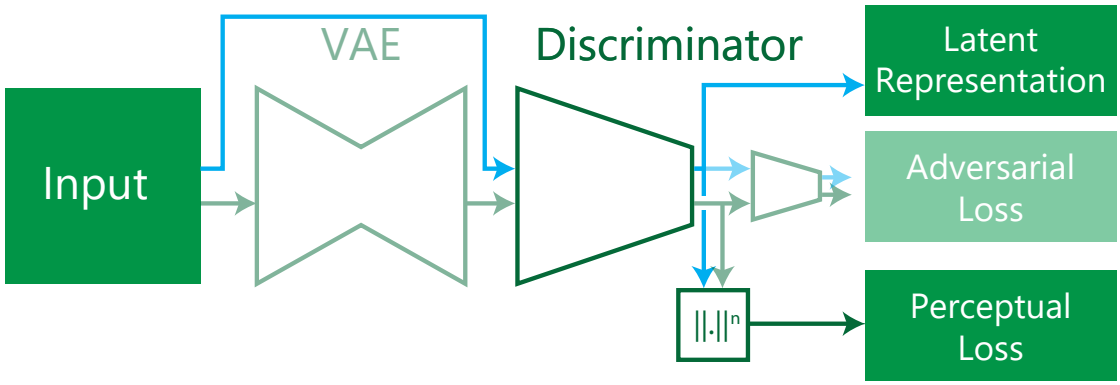


Figure 5.11 – The framing of the perceptual loss trains the VAE, not to simply reconstruct the input by minimizing the distance in the input space, but to produce an output with a similar latent representation. The perceptual loss is derived from the learnt features of the discriminator.

As this work does not use self-supervised learning, it is unclear if it can be as robust as the next methods we will present. However, these methods are not necessarily incompatible, opening the possibility for further improved representations.

#### 5.2.3.2 Bootstrap your own latent (BYOL)

Grill et al. 2020 introduced Bootstrap your own latent (BYOL), a new approach for self-supervised learning of representations for images.

As illustrated in Figure 5.12, BYOL relies on two sub-models: the online and target networks. Each sub-model is itself made of 3 similar parts: 1) an augmentation $t$, 2) an encoder $f$ and 3) a projector $g$. The online network has one additional part which is a predictor $q$.

For each sub-model, the input $x$ is first augmented, yielding a new view $v = t(x)$. This view is then encoded into a representation $y = f(v)$. Afterwards, $y$ is fed to $g$, giving the projection $z = g(y)$. Finally, for the online network, the projection $z$ is given to $q$ which outputs the prediction $q(z)$.

It is important to note the online and target network do not share weights. The weights of the online network are referred to as $\theta$ and those of the target network as $\xi$. Therefore, for example $y_\theta = f_\theta(v)$ refers to the representation obtained by the online network. Additionally, the augmentations are also different and are referred to as $t$ for the online network and as $t'$ for the target network.

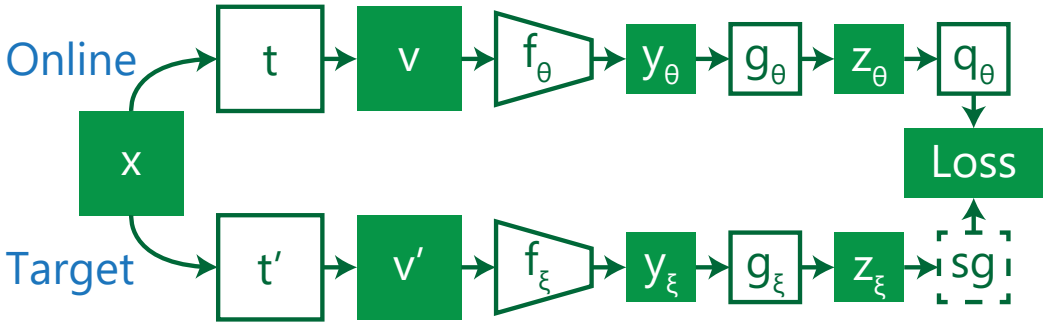


Figure 5.12 – The online and target networks encode and then project different views of the same input. The online network is trained to predict the target network's output. The desired representations are obtained from the encoder of the online network. *sg* designates the stop-gradient operator and prevents gradients from updating the target network.

On one hand, the online network is trained to minimize the distance between its prediction $q_\theta(z_\theta)$ and the output of the target network $z_\xi$. On the other hand, the target network is not trained. Instead, its weights $\xi$ are an exponential moving average of the online network's weights $\theta$.

In similar setups in the literature, the model is also given pairs of inputs where the two inputs are different views of the same input. The model is then asked to minimize the distance between the representation of the same input. However, a trivial solution for the model would be to map all inputs to a single point. To counter this, these models are usually also fed with negative pairs for which they have to maximize the distance between their representation.

However, the setup of BYOL allows it to reach state-of-the-art results, on a downstream classification task on ImageNet, without having to rely on negative pairs as it is most commonly done.

As BYOL was built for images, the augmentations $t$ and $t'$ would have to be adapted in order to use BYOL with other modalities. Additionally, when modeling human behavior, it would be important to consider the conditions $t$ and $t'$ have to meet to preserve information related to the behavior.

#### 5.2.3.3 Self-distillation with no labels (DINO)

Caron et al. 2021 introduced Self-distillation with no labels (DINO), a new approach for self-supervised learning of representations for images similar to BYOL on several points.

DINO is also made of two sub-models, referred to as the teacher and the student. Similarly as to BYOL, the teacher's parameters are an exponential moving average of the student, hence the name "self-distillation". For consistency purposes, we will refer to these parameters in the same way as for BYOL, where $\theta$ designates the student's parameters and $\xi$ the teacher's parameters.

However, as illustrated in Figure 5.13, DINO does not produce two different views of the same input. Instead, the difference occurs after the input has been encoded by both sub-models. After the teacher has encoded the input $x$, the resulting representation $y_\xi = f_\xi(x)$ is, put simply, centered around zero. To be more precise, DINO also keeps track of $ema(y_\xi)$ the exponential moving average of $y_\xi$ and subtracts it from $y_\xi$. Then, both $y_\theta$ and $y_\xi - ema(y_\xi)$ are passed through a softmax operator. Finally, the student is trained to minimize the distance between $softmax(y_\theta)$ and $softmax(y_\xi - ema(y_\xi))$.

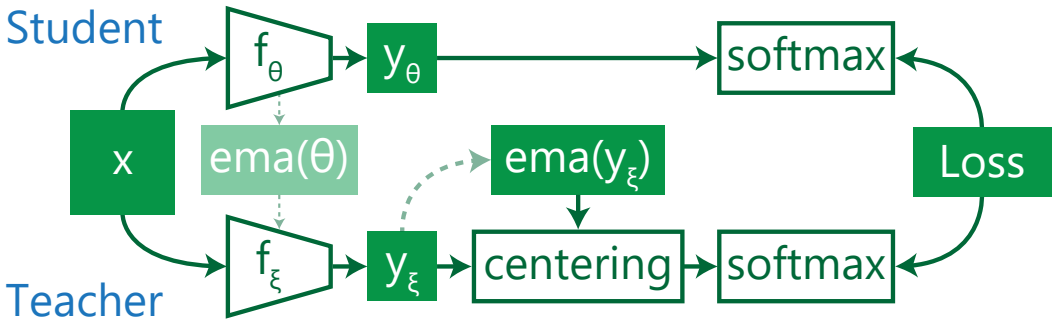


Figure 5.13 – The student and the teacher first encode the same input. Then, the teacher centers its resulting representation, which is then compared to the student's output. Dashed lines represent updates made after a train step.

As with BYOL, DINO reaches state-of-the-art results on a downstream classification task on ImageNet, again without requiring popular components such as negative pairs to avoid collapse.

Additionally, contrarily to BYOL, DINO does not need augmentations to produce new views. This enables DINO to be used on any modality without having to produce new augmentations. In our case, this can be both a good or a bad thing if we consider structural priors as we discussed in Section 5.1.3. Indeed, these augmentations can introduce structural priors. This can either be helpful or harmful to the learning process, depending on the augmentation process we chose.

#### 5.2.3.4 Audio-visual Multisensory Features

Finally, we briefly go over the work of Owens and Efros 2018, as we already covered it in Section 3.2.4.

Owens and Efros introduce a new approach for self-supervised learning of audiovisual representations. They use a contrastive loss — *i.e.* using positive and negative pairs — where the negative pairs are obtained by shifting the audio track of a positive pair in either direction, de-synchronizing it from the video track.

As illustrated in Figure 5.14, the model is composed of 4 parts: One encoder for each modality, one joint encoder and a final fully-connected layer yielding the energy value of the model. Once the model has been trained, the final fully-connected layer is discarded and the representations yielded by the joint encoder are used for the downstream task (*e.g.*

visualize sound sources, action recognition, on/off-screen audiovisual source separation, *etc.*)

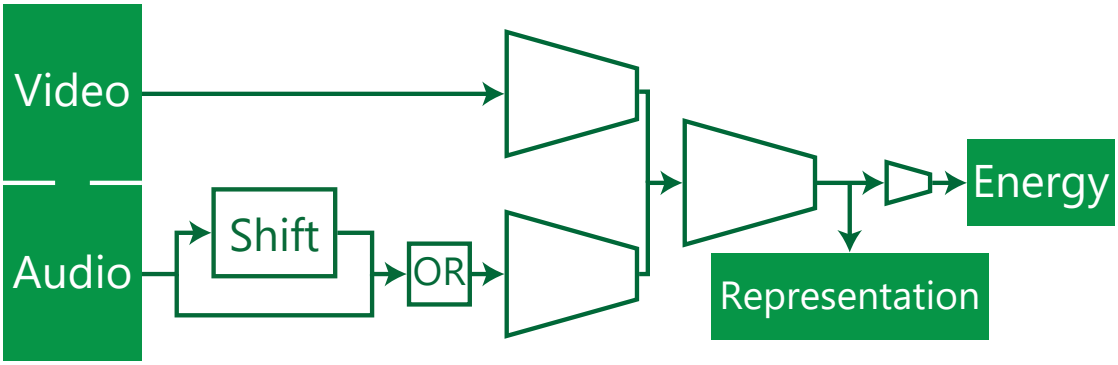


Figure 5.14 – Assuming audio and video tracks from the same sample, both tracks are encoded using two independent encoders. Then, the resulting intermediate representations are concatenated and fed to a joint encoder, yielding the desired representation. During training, the audio track is shifted half the time and the joint representation is fed to a final layer that is used to determine if the audio track was shifted.

This method has two strong points: 1) The method is simple to use, yet the resulting model is powerful and its representations can be used for a varied number of downstream tasks. 2) This is a method for audiovisual data, which is rare enough to be pointed out, and is of particular interest to us.

Additionally, we were able to replicate the work of Owens and Efros, not only on the Audioset dataset as they, but on the Emo&ly as well, indicating this method can be easily transferred to need datasets.

While the shifting of the audio track teaches the model to estimate the temporal consistency between the video and the audio track, further work is probably required to produce representations specific to human behavior.

## 5.3 (Un-)Reproducibility

Reproducibility is essential in experimental fields of science. Even if reproducibility is only one criteria, it is required to trust the experimental findings reported in publications. Fully reproducible works guarantee that the presented methods can achieve the

results provided in their publications. Importantly as well, granting access to the data and software used for the experiments saves time for future users, whether they are trying to replicate the experiments or using it for their own research or trying to use it for production.

We will discuss the problems we faced regarding reproducibility and what we did to ensure our experiments were reproducible as much as possible. We split these problems in three categories: 1) problems related to data, 2) problems related to code (and software) and 3) problems related to the hardware used in experiments.

### 5.3.1 Data

During this thesis, while data was the side we had the least problems regarding reproducibility, we still faced some issues.

While a rather rare issue, publications made by the private sector sometimes used internal datasets which they did not released. Since these datasets were not available, it would not have been possible to replicate the same experiments.

Additionally, we faced some issues with toy datasets some authors used to produce proofs of concept for their method. Access to these toy datasets was not often provided in the paper.

In order to avoid these problems in this thesis, we only used publicly available datasets and listed them in Section 2.2. Additionally, the labels we introduced for the Subway Entrance dataset are available in the github repository of the related paper.

It is worth noting that both the Emo&ly and Subway datasets are not hosted on a public platform and must be requested from their respective authors. Additionally, researchers requesting the Emo&ly dataset must sign an EULA (End User License Agreement).

### 5.3.2 Software

As we showed in Section 2.3.2.2 and Table 2.3, while most experimentation in the literature are conducted on publicly available datasets, very few authors provide their code and even fewer authors provide the resulting parameters — not hyper-parameters — of their model.

#### 5.3.2.1 Attempts at reproducing previous works

While we were able to reproduce methods with low complexity, such as the STAE (Y. Zhao et al. 2017), Deep Image Prior (Ulyanov, Vedaldi, and Lempitsky 2018) or even the work of Owens and Efros on audiovisual representation learning (Owens and Efros 2018), we failed to replicate more complex works such as the Kanerva Machine (Y. Wu, Wayne, Graves, et al. 2018; Y. Wu, Wayne, Gregor, et al. 2018) or the Scaling autoregressive video models (Weissenborn, Täckström, and Uszkoreit 2019).

We cannot be sure if this failure is due to our attempt at replicating the code for these methods, or if it is due to a lack of computational capabilities, or if the findings are not presented in a reproducible way. Nonetheless, we failed at replicating these works, which would not have happened if the tools — data, code and hardware — had been publicly accessible.

#### 5.3.2.2 A framework built for reproducibility and modularity

To avoid having the aforementioned issues in our work, we developed an extensive code base for anomaly detection using deep learning.

This code base takes care of most of the pipeline. First, it pre-processes the datasets in order to improve the speed during training and evaluation. These pre-processes also include extracting pre-computed features (such as the Mel-frequency Cepstral Coefficients (MFCCs) for the audio or the optical flow for the video).

Second, our code base contains all building blocks — used for our models — that do not already exist in the Tensorflow (Martín Abadi et al. 2015) and Keras (Chollet et al. 2015) libraries. Third, based on these building blocks, our framework contains all methods — and models — we developed. Additionally, all methods are defined in a consistent manner. This allows us to only change the method, keeping the rest of the experiment unchanged, ensuring fair comparison between methods.

Then, using a simple configuration file, the main part of our framework selects the model to use and all hyper-parameters (the number of layers, the size and lengths for input videos, *etc.*). Importantly, this configuration file defines a seed for random processes. This seed is used for every single explicit random process in all the pipeline, ensuring the same random numbers are drawn every time as long as the seed remains the same, improving reproducibility by a large margin. However, we cannot control the randomness introduced by the parallel nature of the hardware, introducing slight differences between

two otherwise identical training.

Once the pipeline has built the model, it can either be trained or evaluated. In both cases, a new unique folder is created, which contains a copy of the configuration file as well as copies of various other configuration files such as a json generated by the Keras library. For the training phase, the weights — or parameters — of the model are also saved in this folder at regular intervals.

For the testing phase, the weights used are also copied, for two reasons: 1) to provide a backup and 2) to create easily identified pairs of $(model, results)$. Additionally, the results of the model are also saved in the folder and include both the performance results (area under the ROC curve, *etc.*) and the anomaly scores computed by the model.

Overall, our framework aims at:

1. Eliminating as much randomness as possible between two trainings/tests.
2. Providing the code from start to finish.
3. Providing essential data related to training/tests, such as the configuration and the resulting model's parameters.

### 5.3.3 Hardware

As we said earlier, we failed to replicate works such as the Kanerva Machine (Y. Wu, Wayne, Graves, et al. 2018; Y. Wu, Wayne, Gregor, et al. 2018) or the Scaling autoregressive video models (Weissenborn, Täckström, and Uszkoreit 2019). While this may be related to a bad implementation of these methods, we believe the hardware we used did not meet the minimum requirements in terms of computational capabilities (mostly GPU memory requirements).

We came to this conclusion because, while trying to replicate these works, we had to reduce the width and depth of all layers in these models. Using the original hyper-parameters, the memory required to train the model would exceed our available memory by several orders of magnitude.

While it was tempting to scale up our hardware capabilities by using public infrastructures such as the Jean Zay supercomputer or the IGRIDA computing grid, we felt we would lose more than we would win by doing so.

Indeed, real-time anomaly detection is a field where speed is crucial. A potentially harmful anomaly must be caught before it has done any harm. Additionally, potential users do not have access to supercomputers. One last reason, but not the least, is the fact

that these limitations constrained us to produce simpler models. As we have shown until now in this thesis, there is still enough work to do on simple models before we try to scale them up. Overall, this made our work reproducible with consumer-level computers and graphic cards, which are fairly accessible.

Chapter 6

# Discussion

Now that we have presented the contributions of this thesis, we summarize them and draw some conclusions.

First, we discuss the mismatches we identified between anomaly detection and its real world applications. We also discuss how we would tackle these problems.

Second, we observe that modalities related to human behavior are more complex and diverse than what can be estimated at first sight. We draw some recommendations from there.

Third, we discuss the structural priors of a model, including their positive and negative aspects.

Fourth, we return to the consideration that human behaviors can be modeled as temporal mechanisms and address the implications of this line of looking at the problem.

Fifth, we see that the joint modeling of multiple modalities is only beginning, especially when considering human behavior and anomalous behaviors.

Sixth, we summarize our contributions on deep representation learning and explain why we think it is likely to become mandatory for detecting anomalous human behaviors.

Seventh, we come back on the subject of datasets. We discuss how we have much more data available than the data from datasets for anomaly detection. We also put a warning on using such large amounts of data.

Finally, we discuss our findings relatively to the expected future deployment of anomaly detection methods in the real world.

## 6.1 Anomaly detection and real world applications

During this thesis, we identified two main categories of mismatches between the problems to be solved and the way these problems are tackled.

First, when defining anomaly detection in Section 1.2.2, we warned the reader about the issue that comes up when there is a mismatch between user's expectations about

anomaly detection and what anomaly detection solutions achieve in practice.

The second main category of mismatches, which we discussed partially in Section 1.3.1, is faced when defining the context and what a normal sample is in this context.

#### 6.1.0.1 What anomaly detection can be expected to achieve / What it actually achieves

As we talked in Section 1.2.2, we have to be careful when defining anomalies in an intuitive way. For example, it is easy to mistake anomaly detection for undesirable event detection and to try to define anomalies as undesirable events.

Instead, to tackle this problem, it is important to remember that anomalies are defined by opposition to normal samples. Therefore, only the normal class must be defined in anomaly detection.

#### 6.1.0.2 What we think is normal for the context / What it defines as normal

As we explained in Section 1.2.1.1, a sample can only be defined as normal or anomalous relatively to a context. This context itself should represent normality.

However, it is important not to fall in the trap of believing the context known by the anomaly detector is the same as the context we want to be modeled. In particular, as we discussed in Section 1.3.1, not all normal behaviors are equally captured in the data that defines our context. In the case of anomalous human behaviors, there will likely be under-represented behaviors missing from the dataset, despite them being normal.

Said otherwise, when constituting a context of normal samples, to tackle this problem, it is important to make sure the under-representation of some normal behaviors is compensated in the dataset.

## 6.2 There is more than meets the eye

As we just saw in Section 6.1, defining a context is not obvious. However, it is not only about gathering more data or accounting for under-representation. It is also about gathering more modalities.

In Section 1.4, we showed there are a wide variety of modalities human behaviors can be observed through. Additionally, there are probably more modalities for that than we listed and we should try to gather multiple modalities describing the same behaviors.

However, these modalities are not the only ones to be considered. While the modalities we mentioned in Section 1.4 are essential, modalities giving contextual information not contained in other modalities are essential.

Let us take the example of public video surveillance. To our knowledge, no dataset for anomaly detection in surveillance video contains any contextual information such as the date or even the period of the year. Depending on the situation, other modalities — such as the time — can be crucial to differentiate normal behaviors from anomalous behaviors.

A very concrete example, where such information would be needed, happened during this thesis: social distancing. For a model trained on data from the pre-covid19 crisis, social distancing would be an anomalous behavior. However, this behavior has become common in the last two years and therefore normal.

For these reasons, we believe that anomaly detection systems that will be deployed in the future should use at least two modalities, where at least one modality should contain contextual information such as the ones we described above.

## 6.3 To prior, or not to prior ?

We now come back on the subject of priors about the data in anomaly detection models. More precisely, on structural priors present even before the model has started training. Depending on the situation, these priors can either be useful or harmful to the performance of the model.

Having good priors allows a model to converge faster, reducing the costs of training — including time, amounts of data, computational power, *etc.*— and helping to find a better solution at first.

This is particularly the case with what Morais et al. 2019 did for example: they automatically extracted skeletons and their trajectories from the video stream, drastically reducing the input space while only keeping relevant information (but not all relevant information). Using skeletons trajectories, they were able to significantly improve over previous methods developed for anomaly detection on surveillance video. However, their method requires that skeletons can be computed from the input, which is not always going to be case: the skeleton detector can fail, cannot be used on face-camera videos and cannot be used on other modalities besides video. While improving performance in specific situation, such priors can be detrimental to broader situations.

In this thesis, we almost only used CNNs, which were showed to biased as well. Since

we are studying anomalous behavior, these biases may always be helpful but this remains to be further studied.

However, what we can do for now is keep in mind that such priors exist. They are not bad by essence and, in the contrary, can prove to be quite useful. Therefore, moving forward, we should aim at building methods and models where those priors have been taken into account and used in a beneficial way.

## 6.4 Modeling behaviors as temporal mechanisms

Human behavior is, by essence, better observed in temporal data. While non-temporal data, such as images, can give clues about the observed behavior, the temporal equivalent, such as videos, always seem to give more clues.

This is because human behaviors are essentially temporal mechanisms. The success of the IAE — described in Section 4.2 — indirectly shows this phenomenon, as it significantly outperforms an autoencoder in an otherwise identical situation, where even the hyper-parameters are the same. This leaves the interpolation during training — and therefore the temporal constraint — the only difference between the two models and that difference seems to fully account for the improvement.

The IAE can be seen as adding strong structural priors about temporal sequences, such as the fact that frames — or blocks of frames — are separated with a constant amount of time. As we discussed in Section 6.3, such priors can be helpful — even essential — to improve a model's performances.

We draw two conclusions from this:

1. Human behaviors should be modeled as temporal mechanisms.
2. Priors such as the ones introduced by the IAE are essential to model behaviors as temporal mechanisms.

Additionally, we add that such mechanism can be interpreted as an invisible cause. This cause can not be directly observed but its consequences may. Likewise, a person's behavior can only be observed through the sequence of actions this person performs. Therefore, identifying a behavior becomes similar to identifying a cause, a mechanism generating observables.

## 6.5 Multimodality is only beginning

When reviewing previous works related to anomalous human behavior detection, we saw that the literature on modeling multiple modalities in a joint manner is still very young. This is notably demonstrated by the following points:

1. Among all the works we listed in Sections 2.1 and 2.3.2, none of them used more than one modality. All works using multiple modalities we found were either not related to anomaly detection — *e.g.* works on representation learning — or not related to human behavior (they were mostly related to assistive robots and device failure).
2. As shown in Table 4.5, methods using pre-computed features such as optical flow or skeletons — or MFCCs for audio — tend to perform better than fully end-to-end methods. However, these pre-computed features cannot be transferred to other modalities. This implies that, currently, monomodal methods will outperform multimodal methods if they use such pre-computed features.
3. To our knowledge, the work of Fayet 2018 is the only work on multimodal anomalous behavior detection. However, even in this case, anomaly detection is performed through a late-fusion of the different modalities. This means the modalities were not modeled jointly, which we believed to be crucial.

However, it is hard to develop and evaluate a method for multimodal anomalous behavior detection when there are no complete dataset for this purpose, the closest being the Emo&ly dataset (Fayet et al. 2018).

To our understanding, a dataset for multimodal anomalous behavior detection should match the following criteria:

1. Be multimodal and preferably use challenging modalities such as audiovisual data.
2. For the reasons we highlighted in Section 2.3.1.2, normality in the dataset should be well-defined by a rule leaving as little room as possible for interpretation. The labels from experts should be consensual and come from several experts.
3. Be, as much as reasonably possible, representative of the challenges posed by the real world. This criteria can also be achieved by building multiple datasets, each with their own challenges.

To summarize, datasets and methods for multimodal anomalous behavior detection are still in early development and fitting datasets will be required before the methods can

be properly evaluated.

## 6.6 Moving forward with Deep Representation Learning

We concluded that the need to develop datasets for multimodal anomalous behavior detection is critical in order to evaluate methods for the same task. However, there are other leads we can pursue in parallel. One of them is deep representation learning.

As we have seen in Section 5.2, deep representation learning allows to partially — or sometimes fully — solve some lingering problems with multimodal anomalous behavior detection (such as those created by comparing modalities in their input spaces, the abstract nature of human behavior and the common lack of interpretability).

Additionally, we also saw in Section 5.2.3 that methods for representation learning are usually simple, which is helpful for reproducing them and makes them reliable.

When it comes to learning representation for human behaviors, there are no model explicitly developed with that goal in mind. However, we can still see how we can use existing methods for representation learning and/or adapt them using priors we have on human behaviors.

## 6.7 We have a lot of unlabeled multimodal data (let's use it)

Similarly to most deep learning models, deep representation models need datasets to be trained on. While this is currently a problem for anomalous behavior detection, it is less of a problem for deep representation learning.

There are plenty of multimodal datasets describing human behaviors that are publicly available. In particular, as we have seen in Section 2.2.3.1, the Audioset dataset (Gemmeke et al. 2017) provides large amounts of audiovisual data, including close to 3 hours of data labeled as "Speech" and 5500 hours overall. To this end, identifying data describing human behaviors is all we would need to start learning representations of human behaviors. The modalities in the data themselves do not matter as long as they describe human behaviors. With data describing human behavior, it becomes possible to extract correlations only

found in this kind of data and therefore learn these correlations in order to model human behavior.

However, an important warning should be given about the datasets chosen for learning representations. Indeed, these datasets may not be themselves representative of the real world and can embed undesirable priors on human behaviors.

This issue is related to the concerns raised by Bender et al. 2021 on large language models. The models they studied seemed to act like parrots. While this may not seem as an issue at first when it comes to learning deep representation models, it is important to note that these language models are not simply repeating training data, they are notably repeating dangerous and harmful ideas.

In our case, this could translate to under-presented behaviors being marginalized in the representations obtained, resulting in higher likelihood for these behaviors to be considered anomalous in a downstream anomaly detection task.

Overall, we think there is still an untapped potential in multimodal datasets to learn representations of human behaviors and we should be careful when learning these representations.

## 6.8 Moving to production

We finish this discussion with a word on different methods that will be useful when it comes to deploying a model in production.

For reasons we highlighted in Section 2.3.3, an anomaly detection system which will be put in production on a long period must be resilient to concept drift. Active learning — which we presented in the same section — and meta-learning models — which we presented in Sections 3.2.5 and 5.1.2 — can be solutions to adapt to such concept drift.

Additionally, an anomaly detection system based on a well-trained meta-learning model will likely allow for quicker and cheaper deployment to new context, as the model will only need a few training samples and a shorter learning phase to adapt to the new context.

Overall, methods that allow for a real-time detection of anomalous behaviors on accessible hardware are more likely to be used in real-world applications. Therefore, while detection performances are important, it is desirable and possible to develop real-time counterparts of slow methods at a small anomaly detection performance cost.

Chapter 7

# Future Works

We now address about the future works we would expect to be investigated in order to improve the detection of anomalous human behaviors. For each focus of these future works, we also bring up potential improvements we foresee for other domains.

First, we discuss the works we expect to see on representation learning as we think it will be pivotal to better anomalous human behavior detection.

Second, we return on datasets and describe what angles we would expect future datasets to cover.

Lastly, we describe improvements we would expect for enabling and evaluating active learning for anomalous behavior detectors.

## 7.1 Representation Learning

As we have seen throughout this thesis, representation learning is a promising field for a broad range of machine learning methods, including the detection of anomalous human behaviors. We also saw that, despite only having drawn attention recently, the field of deep representation learning already saw plenty of work being made.

We anticipate work on representation learning for three different tasks:

1. Simply improving the internal representations of real-world concepts — abstract or not — for various reasons.

2. Improving the interpretability of future models by creating relations between the representations and their corresponding real-world concepts.

3. Bridging the gap between the fields of representation learning, human behavior and anomaly detection.

### 7.1.1 Improving the modeling of real-world concepts in latent spaces

The future works on representation learning we expect the most will improve the consistency between the latent representations learnt by the model and their matching real-world concepts.

Works like those we mentioned in Section 5.2.3 (Owens and Efros 2018; Larsen et al. 2016; Grill et al. 2020; Caron et al. 2021) already pave the way and there is still plenty of work to be done.

Notably, we expect some future works to study the relation between the method being used for representation learning — *e.g.* VAE-GANs, BYOL, DINO, *etc.*— and the way these methods condition the learnt representation. For example, how does the method introduced by Owens and Efros 2018 affects the learnt representation of audiovisual data ? Conversely, and more importantly to us, what method should we use to condition the model to focus on representing human behaviors ?

### 7.1.2 Interpretability through related representations

The methods and models we described so far focus on mapping real-world concepts to a latent representation space. However, to our knowledge, there are currently no method developed specifically to do the opposite: going from the latent space to the matching real-world concept space.

Therefore, we expect future works to be made with this goal in mind, as this could improve the interpretability of decision models such as anomaly detectors. For example, an anomaly detector could map the representation of its decision to a modality understandable by humans using a backward representation model. In critical situations, we expect this could help a human operator to make the final decision. Additionally, this may help improve the confidence of the final user in the model if its decision can be explained.

### 7.1.3 Anomaly detection and representation learning

Finally, the anomaly detection and representation learning fields are currently mostly evolving independently. As we expressed several times already, we believe anomaly detection — especially the detection of anomalous human behaviors — can benefit from representation but the opposite could be also true.

While not expecting much future works being made on this subject, we could foresee some work being made to improve representation learning thanks to anomaly detection. For example, an anomaly detection method could be used to constrain a representation model to avoid learning anomalous representation (in the sense of representation with a particularly low likelihood). Such a framework would be designed to prevent a representation model to learn multiple representations of noise — improving robustness — and instead learn a single one, basically labeled "noise".

Nonetheless, we mostly expect work to be made to improve anomaly detection using representation learning, with all the advantages we have listed in this thesis.

## 7.2 Future datasets

During this thesis, it was clear that new datasets are required for multimodal anomalous human behavior detection. This is what motivated Fayet to build the Emo&ly dataset 2.5. However, there is still plenty of work to be done in this area before we can begin to properly evaluate methods for multimodal anomalous behavior detection.

We foresee four areas to work on for building the datasets we need: 1) benchmarks for representation learning methods 2) representative datasets 3) challenging datasets and 4) privacy-preserving datasets.

### 7.2.1 Representation learning benchmarks

On representation learning, we expect to see datasets that include a way to investigate — or even evaluate — the representations learnt by a model. This would notably help developing the potential works on representation learning we discussed in Section 7.1, for example by making sure the representations learnt match actual real-world concepts and not some unhelpful patterns.

### 7.2.2 Representativity of a dataset

When building new datasets, future works should make sure the dataset is representative and not simply a biased representation of reality. We bring this point as an echo of the issues we presented in Section 1.3.1.

While the immediate effect of taking care of representativity might be an extra cost in time, the long term effect of a non-representative dataset are likely to be worse on

the long run. Real-world cases of such imbalance were already witnessed and negatively affected millions of users (Yee, Tantipongpipat, and Mishra 2021; Obermeyer et al. 2019).

There is also the vague possibility of future works to be conducted on automatically evaluating the representativity of a dataset, or even automatically updating a non-representative dataset into a representative one. However, such method would have to be evaluated manually itself.

### 7.2.3 Challenges posed by a dataset

While not all datasets should be very challenging, there is a need for datasets that mimic the real-world challenges in order to verify if a method can safely be deployed in the real world.

To that extent, futures works could address the need for a dataset with multiple levels or challenges. This would allow methods in early development to be tested on easier challenges — *i.e.* for a proof of concept — while more mature methods would be evaluated on a more complete spectrum of challenges. Another added benefit of such a dataset would be to assess the strengths and — more importantly — weaknesses of a method, yielding leads to improve the method.

### 7.2.4 Privacy-preserving datasets

Privacy-preserving machine learning is a method for significantly increasing the amount of data available while not compromising privacy (K. Xu et al. 2015). Privacy-preserving also has been developed for deep learning (Ryffel et al. 2018).

While privacy-preserving machine learning is not simply a way to build datasets, this is a related topic we think will attract a lot of attention in the future. The base concept is the following: allowing a model to be trained on potentially sensitive data while letting as little private information leak as possible. This can typically be done by having the organisation owning the data train the model on their own servers and only yield the necessary information about the training.

As we explained at the very beginning of this manuscript in Section 1.1.2, contexts where one can want to detect anomalous human behaviors are far to be limited to video surveillance. These contexts notably include the mental health field and the field of detecting invisible disabilities. The information in these fields can be very sensitive. In our opinion, if we ever intend to build helpful models for such contexts, privacy-preserving ma-

chine learning is going to be mandatory for the development of these anomalous behavior detectors.

## 7.3 Adaptive methods

We finish the list of future works we expect with methods that should be developed to better adapt to the dynamic and changing nature of the real world.

These futures work concern two fields we already talked about: Meta-learning — in Sections 2.3.3, 3.2.5 and 6.8 — and Active learning, which we discussed in Sections 2.3.3 and 6.8.

### 7.3.1 Future works on meta-learning

Overall, the applications of meta-learning are close to limitless. Indeed, meta-learning models learn to learn. Therefore, they can potentially be used in any machine learning application and we expect their interest will grow as the field of deep learning is explored. One could argue that any reasonable problem could be solved by an agent able to teach itself anything.

This means that, in our opinion, anomaly detection will also benefit from meta-learning. Not only to improve the performance of anomaly detectors, but also improve their transferability on new contexts. Additionally, we expect meta-learning models for anomaly detection will be studied to reduce the amount of labeled data in the frame of semi-supervised learning where, basically, all anomalies are supposed to be identified.

### 7.3.2 Future works on active learning

While the literature on active anomaly detection has been developed to allow anomaly detection models to receive feedback and improve based on this feedback, to our knowledge, this does not concern deep learning models. This is because deep learning are trained over at least tens of thousands of iterations, sometimes millions of iterations or even more, where each sample is seen a significant amount of time. Still to our knowledge, there are no way to incorporate a few elements in the dataset without having to retrain the whole model.

We therefore expect futures works to be conducted on this problem in order to allow deep learning models for anomaly detection to quickly adapt to feedback and potentially

concept drift. While we expect meta-learning models to be well-suited for this job, it is not excluded that other types of models may be as well suited, if not more.

# Primary references


Abati, Davide et al. (2019), « Latent Space Autoregression for Novelty Detection », *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 481–490.

Adam, Amit et al. (2008), « Robust real-time unusual event detection using multiple fixed-location monitors », *in*: *IEEE transactions on pattern analysis and machine intelligence* 30.*3*, pp. 555–560.

Agarap, Abien Fred (2018), « Deep learning using rectified linear units (relu) », *in*: *arXiv preprint arXiv:1803.08375*.

An, Jinwon and Sungzoon Cho (2015), « Variational autoencoder based anomaly detection using reconstruction probability », *in*: *Special Lecture on IE* 2.*1*, pp. 1–18.

Andrews, Jerone, Edward Morton, and Lewis Griffin (Jan. 2016), « Detecting anomalous data using auto-encoders », *in*: *International Journal of Machine Learning and Computing* 6, p. 21.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017), *Wasserstein GAN*, arXiv: `1701.07875 [stat.ML]`.

Bachman, Philip, R Devon Hjelm, and William Buchwalter (2019), « Learning representations by maximizing mutual information across views », *in*: *arXiv preprint arXiv:1906.00910*.

Breunig, Markus M et al. (2000), « LOF: identifying density-based local outliers », *in*: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104.

Caron, Mathilde et al. (2021), « Emerging properties in self-supervised vision transformers », *in*: *arXiv preprint arXiv:2104.14294*.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2009), « Anomaly detection: A survey », *in*: *ACM computing surveys (CSUR)* 41.*3*, p. 15.

Chen, Yuting, Jing Qian, and Venkatesh Saligrama (2013), « A new one-class SVM for anomaly detection », *in*: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, pp. 3567–3571.

Chong, Yong Shean and Yong Haur Tay (2017), « Abnormal event detection in videos using spatiotemporal autoencoder », *in*: *International symposium on neural networks*, Springer, pp. 189–196.

De, Soham and Samuel L Smith (2020), « Batch normalization biases residual blocks towards the identity function in deep networks », *in*: *arXiv preprint arXiv:2002.10444*.

De Vries, Timothy, Sanjay Chawla, and Michael E Houle (2010), « Finding local anomalies in very high dimensional space », *in*: *2010 IEEE International Conference on Data Mining*, IEEE, pp. 128–137.

Durand de Gevigney, Valentin et al. (2020), « Video Latent Code Interpolation for Anomalous Behavior Detection », *in*: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, pp. 3037–3044.

Erfani, Sarah M et al. (2016), « High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning », *in*: *Pattern Recognition* 58, pp. 121–134.

Fan, Yaxiang et al. (2018), « Video anomaly detection and localization via Gaussian mixture fully convolutional variational autoencoder », *in*: *arXiv preprint arXiv:1805.11223*.

Fayet, Cédric (2018), « Multimodal anomaly detection in discourse using speech and facial expressions », PhD thesis, Rennes 1.

Fayet, Cédric et al. (2018), « EMO&LY (EMOtion and AnomaLY): A new corpus for anomaly detection in an audiovisual stream with emotional context. », *in*: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Feng, Yachuang, Yuan Yuan, and Xiaoqiang Lu (2017), « Learning deep event models for crowd anomaly detection », *in*: *Neurocomputing* 219, pp. 548–556.

Flach, Peter A and Meelis Kull (2015), « Precision-Recall-Gain Curves: PR Analysis Done Right. », *in*: *NIPS*, vol. 15.

Foggia, Pasquale et al. (2015), « Audio surveillance of roads: A system for detecting anomalous sounds », *in*: *IEEE transactions on intelligent transportation systems* 17.*1*, pp. 279–288.

Gemmeke, Jort F. et al. (2017), « Audio Set: An ontology and human-labeled dataset for audio events », *in*: *Proc. IEEE ICASSP 2017*, New Orleans, LA.

Goodfellow, Ian et al. (2014), « Generative adversarial nets », *in*: *Advances in neural information processing systems* 27.

Graves, Alex, Greg Wayne, and Ivo Danihelka (2014), « Neural turing machines », *in*: *arXiv preprint arXiv:1410.5401*.

Graves, Alex, Greg Wayne, Malcolm Reynolds, et al. (2016), « Hybrid computing using a neural network with dynamic external memory », *in*: *Nature* 538.*7626*, pp. 471–476.

Grill, Jean-Bastien et al. (2020), « Bootstrap your own latent: A new approach to self-supervised learning », *in*: *arXiv preprint arXiv:2006.07733*.

Hariri, Sahand, Matias Carrasco Kind, and Robert J. Brunner (2019), « Extended Isolation Forest », *in*: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, ISSN: 2326-3865, DOI: 10.1109/tkde.2019.2947676, URL: http://dx.doi.org/10.1109/TKDE.2019.2947676.

Hasan, Mahmudul et al. (2016), « Learning temporal regularity in video sequences », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 733–742.

Hawkins, Douglas M (1980), *Identification of outliers*, vol. 11, Springer.

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.

Hinami, Ryota, Tao Mei, and Shin'ichi Satoh (2017), « Joint detection and recounting of abnormal events by learning deep generic knowledge », *in*: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3619–3627.

Hu, Di et al. (2020), « Discriminative sounding objects localization via self-supervised audiovisual matching », *in*: *Advances in Neural Information Processing Systems* 33.

Kim, Jaechul and Kristen Grauman (2009), « Observe locally, infer globally: a space-time MRF for detecting abnormal activities with incremental updates », *in*: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 2921–2928.

Kingma, Diederik P and Max Welling (2013), « Auto-encoding variational bayes », *in*: *arXiv preprint arXiv:1312.6114*.

Larsen, Anders Boesen Lindbo et al. (2016), « Autoencoding beyond pixels using a learned similarity metric », *in*: *International conference on machine learning*, PMLR, pp. 1558–1566.

LeCun, Yann, Sumit Chopra, et al. (2006), « A tutorial on energy-based learning », *in*: *Predicting structured data* 1.*0*.

Li, Kun-Lun et al. (2003), « Improving one-class SVM for anomaly detection », *in*: *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693)*, vol. 5, IEEE, pp. 3077–3081.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008), « Isolation forest », *in*: *2008 eighth ieee international conference on data mining*, IEEE, pp. 413–422.

Liu, Ming-Yu, Thomas Breuel, and Jan Kautz (2017), « Unsupervised image-to-image translation networks », *in*: *Advances in neural information processing systems*, pp. 700–708.

Liu, Wen et al. (2018), « Future frame prediction for anomaly detection–a new baseline », *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6536–6545.

Lu, Cewu, Jianping Shi, and Jiaya Jia (2013), « Abnormal event detection at 150 fps in matlab », *in*: *Proceedings of the IEEE international conference on computer vision*, pp. 2720–2727.

Luo, Weixin, Wen Liu, and Shenghua Gao (2017), « A revisit of sparse coding based anomaly detection in stacked rnn framework », *in*: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 341–349.

MacKay, David JC and David JC Mac Kay (2003), *Information theory, inference and learning algorithms*, Cambridge university press.

Mahadevan, Vijay et al. (2010), « Anomaly detection in crowded scenes », *in*: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 1975–1981.

Marchi, Erik et al. (2015), « A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks », *in*: *Proceedings 40th IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2015*, 5–pages.

Marteau, Pierre-François (2021), « Random Partitioning Forest for Point-Wise and Collective Anomaly Detection—Application to Network Intrusion Detection », *in*: *IEEE Transactions on Information Forensics and Security* 16, pp. 2157–2172.

Mathieu, Michael, Camille Couprie, and Yann LeCun (2015), « Deep multi-scale video prediction beyond mean square error », *in*: *arXiv preprint arXiv:1511.05440*.

Mehran, Ramin, Alexis Oyama, and Mubarak Shah (2009), « Abnormal crowd behavior detection using social force model », *in*: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 935–942.

Mikolov, Tomas et al. (2013), « Efficient estimation of word representations in vector space », *in*: *arXiv preprint arXiv:1301.3781*.

Morais, Romero et al. (2019), « Learning regularity in skeleton trajectories for anomaly detection in videos », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11996–12004.

Nedelkoski, Sasho, Jorge Cardoso, and Odej Kao (2019), « Anomaly detection from system tracing data using multimodal deep learning », *in*: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, IEEE, pp. 179–186.

Owens, Andrew and Alexei A Efros (2018), « Audio-visual scene analysis with self-supervised multisensory features », *in*: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 631–648.

Owens, Andrew, Jiajun Wu, et al. (2018), « Learning sight from sound: Ambient sound provides supervision for visual learning », *in*: *International Journal of Computer Vision* 126.*10*, pp. 1120–1137.

Park, Daehyung, Zackory Erickson, et al. (2016), « Multimodal execution monitoring for anomaly detection during robot manipulation », *in*: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 407–414.

Park, Daehyung, Yuuna Hoshi, and Charles C Kemp (2018), « A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder », *in*: *IEEE Robotics and Automation Letters* 3.*3*, pp. 1544–1551.

Park, Daehyung, Hokeun Kim, and Charles C Kemp (2019), « Multimodal anomaly detection for assistive robots », *in*: *Autonomous Robots* 43.*3*, pp. 611–629.

Ravanbakhsh, Mahdyar, Moin Nabi, et al. (2018), « Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection », *in*: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, pp. 1689–1698.

Ravanbakhsh, Mahdyar, Enver Sangineto, et al. (2019), « Training adversarial discriminators for cross-channel abnormal event detection in crowds », *in*: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, pp. 1896–1904.

Rodrigues, Royston et al. (2019), « Multi-timescale Trajectory Prediction for Abnormal Human Activity Detection », *in*: *arXiv preprint arXiv:1908.04321*.

— (Mar. 2020), « Multi-timescale Trajectory Prediction for Abnormal Human Activity Detection », *in*: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Rushe, Ellen and Brian Mac Namee (2019), « Anomaly detection in raw audio using deep autoregressive networks », *in*: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 3597–3601.

Sabokrou, Mohammad, Mahmood Fathy, and Mojtaba Hoseini (2016), « Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder », *in*: *Electronics Letters* 52.*13*, pp. 1122–1124.

Sabokrou, Mohammad, Mohsen Fayyaz, et al. (2018), « Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes », *in*: *Computer Vision and Image Understanding* 172, pp. 88–97.

Sakurada, Mayu and Takehisa Yairi (2014), « Anomaly detection using autoencoders with nonlinear dimensionality reduction », *in*: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4–11.

Santoro, Adam et al. (2016), « Meta-learning with memory-augmented neural networks », *in*: *International conference on machine learning*, PMLR, pp. 1842–1850.

Saurav, Sakti et al. (2018), « Online anomaly detection with concept drift adaptation using recurrent neural networks », *in*: *Proceedings of the acm india joint international conference on data science and management of data*, pp. 78–87.

Schölkopf, Bernhard et al. (1999), « Support vector method for novelty detection. », *in*: *NIPS*, vol. 12, Citeseer, pp. 582–588.

Smaira, Lucas et al. (2020), « A short note on the kinetics-700-2020 human action dataset », *in*: *arXiv preprint arXiv:2010.10864*.

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky (2018), « Deep image prior », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454.

Vaswani, Ashish et al. (2017), « Attention is all you need », *in*: *Advances in neural information processing systems*, pp. 5998–6008.

Wang, Yanxin, Johnny Wong, and Andrew Miner (2004), « Anomaly intrusion detection using one class SVM », *in*: *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, pp. 358–364.

Weissenborn, Dirk, Oscar Täckström, and Jakob Uszkoreit (2019), « Scaling autoregressive video models », *in*: *arXiv preprint arXiv:1906.02634*.

Wu, Yan, Greg Wayne, Alex Graves, et al. (2018), « The kanerva machine: A generative distributed memory », *in*: *arXiv preprint arXiv:1804.01756*.

Xu, Dan et al. (2017), « Detecting anomalous events in videos by learning deep representations of appearance and motion », *in*: *Computer Vision and Image Understanding* 156, pp. 117–127.

Yousefi-Azar, Mahmood et al. (2017), « Autoencoder-based feature learning for cyber security applications », *in*: *2017 International joint conference on neural networks (IJCNN)*, IEEE, pp. 3854–3861.

Zambon, Daniele, Cesare Alippi, and Lorenzo Livi (2018), « Concept drift and anomaly detection in graph streams », *in*: *IEEE transactions on neural networks and learning systems* 29.*11*, pp. 5592–5605.

Zhang, Hongyi, Yann N Dauphin, and Tengyu Ma (2019), « Fixup initialization: Residual learning without normalization », *in*: *arXiv preprint arXiv:1901.09321*.

Zhang, Rui et al. (2007), « One class support vector machine for anomaly detection in the communication network performance data », *in*: *Proceedings of the 5th conference on Applied electromagnetics, wireless and optical communications*, Citeseer, pp. 31–37.

Zhao, Hang et al. (2016), « Loss functions for image restoration with neural networks », *in*: *IEEE Transactions on computational imaging* 3.*1*, pp. 47–57.

Zhao, Junbo, Michael Mathieu, and Yann LeCun (2016), « Energy-based generative adversarial network », *in*: *arXiv preprint arXiv:1609.03126*.

Zhao, Yiru et al. (2017), « Spatio-temporal autoencoder for video anomaly detection », *in*: *Proceedings of the 25th ACM international conference on Multimedia*, ACM, pp. 1933–1941.

Zhou, Yipin et al. (2018), « Visual to sound: Generating natural sound for videos in the wild », *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3550–3558.

# Secondary references


Bagirathan, Anandhi et al. (2021), « Recognition of positive and negative valence states in children with autism spectrum disorder (ASD) using discrete wavelet transform (DWT) analysis of electrocardiogram signals (ECG) », *in*: *Journal of Ambient Intelligence and Humanized Computing* 12.*1*, pp. 405–416.

Beltagy, Iz, Matthew E Peters, and Arman Cohan (2020), « Longformer: The long-document transformer », *in*: *arXiv preprint arXiv:2004.05150*.

Bender, Emily M et al. (2021), « On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? », *in*: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623.

Bosl, William J, Helen Tager-Flusberg, and Charles A Nelson (2018), « EEG analytics for early detection of autism spectrum disorder: a data-driven approach », *in*: *Scientific reports* 8.*1*, pp. 1–20.

Brewer, Paul R (2014), « Public opinion about gay rights and gay marriage », *in*: *International Journal of Public Opinion Research* 26.*3*, pp. 279–282.

Chollet, François et al. (2015), *Keras*, `https://keras.io`.

Choromanski, Krzysztof et al. (2020), « Rethinking attention with performers », *in*: *arXiv preprint arXiv:2009.14794*.

Devlin, Jacob et al. (2018), « Bert: Pre-training of deep bidirectional transformers for language understanding », *in*: *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, Alexey, Lucas Beyer, et al. (2020), « An image is worth 16x16 words: Transformers for image recognition at scale », *in*: *arXiv preprint arXiv:2010.11929*.

Dosovitskiy, Alexey, Philipp Fischer, et al. (2015), « Flownet: Learning optical flow with convolutional networks », *in*: *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766.

Fisher, Ronald A (1936), « The use of multiple measurements in taxonomic problems », *in*: *Annals of eugenics* 7.*2*, pp. 179–188.

Gulrajani, Ishaan et al. (2017), « Improved training of wasserstein gans », *in*: *arXiv preprint arXiv:1704.00028*.

Hoang, Lê Nguyên (2021), « Tournesol: Collaborative content recommendations », *in*: *Tournesol.app*, URL: `https://bit.ly/tournesol-app`.

Horn, Berthold KP and Brian G Schunck (1981), « Determining optical flow », *in*: *Artificial intelligence* 17.*1-3*, pp. 185–203.

Ilg, Eddy et al. (2017), « Flownet 2.0: Evolution of optical flow estimation with deep networks », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470.

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015), « Human-level concept learning through probabilistic program induction », *in*: *Science* 350.*6266*, pp. 1332–1338.

Lau-Zhu, Alex, Michael PH Lau, and Gráinne McLoughlin (2019), « Mobile EEG in research on neurodevelopmental disorders: Opportunities and challenges », *in*: *Developmental cognitive neuroscience* 36, p. 100635.

LeCun, Yann, Bernhard Boser, et al. (1989), « Handwritten digit recognition with a back-propagation network », *in*: *Advances in neural information processing systems* 2.

LeCun, Yann, Léon Bottou, et al. (1998), « Gradient-based learning applied to document recognition », *in*: *Proceedings of the IEEE* 86.*11*, pp. 2278–2324.

Limaye, Hrishikesh and VV Deshmukh (2016), « ECG noise sources and various noise removal techniques: a survey », *in*: *International Journal of Application or Innovation in Engineering & Management* 5.*2*, pp. 86–92.

Ma, Minghua et al. (2018), « Robust and rapid adaption for concept drift in software system anomaly detection », *in*: *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, pp. 13–24.

Martín Abadi et al. (2015), *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, URL: `https://www.tensorflow.org/`.

Obermeyer, Ziad et al. (2019), « Dissecting racial bias in an algorithm used to manage the health of populations », *in*: *Science* 366.*6464*, pp. 447–453.

Oord, Aaron van den et al. (2016), « Wavenet: A generative model for raw audio », *in*: *arXiv preprint arXiv:1609.03499*.

Prince, Emily Barbara et al. (2017), « The relationship between autism symptoms and arousal level in toddlers with autism spectrum disorder, as measured by electrodermal activity », *in*: *Autism* 21.*4*, pp. 504–508.

Provost, Foster and Tom Fawcett (2001), « Robust classification for imprecise environments », *in*: *Machine learning* 42.*3*, pp. 203–231.

Radford, Alec et al. (2019), « Language models are unsupervised multitask learners », *in*: *OpenAI blog* 1.*8*, p. 9.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015), « U-net: Convolutional networks for biomedical image segmentation », *in*: *International Conference on Medical image computing and computer-assisted intervention*, Springer, pp. 234–241.

Ryffel, Theo et al. (2018), « A generic framework for privacy preserving deep learning », *in*: *arXiv preprint arXiv:1811.04017*.

Sameni, Reza et al. (2007), « Multichannel ECG and noise modeling: Application to maternal and fetal ECG signals », *in*: *EURASIP Journal on Advances in Signal Processing* 2007, pp. 1–14.

Skinner, Burrhus Frederic (1965), *Science and human behavior*, 92904, Simon and Schuster.

Van Oord, Aaron, Nal Kalchbrenner, and Koray Kavukcuoglu (2016), « Pixel recurrent neural networks », *in*: *International Conference on Machine Learning*, PMLR, pp. 1747–1756.

Wang, Zhou, Alan C Bovik, et al. (2004), « Image quality assessment: from error visibility to structural similarity », *in*: *IEEE transactions on image processing* 13.*4*, pp. 600–612.

Wang, Zhou, Ligang Lu, and Alan C Bovik (2004), « Video quality assessment based on structural distortion measurement », *in*: *Signal processing: Image communication* 19.*2*, pp. 121–132.

Wu, Yan, Greg Wayne, Karol Gregor, et al. (2018), « Learning attractor dynamics for generative memory », *in*: *arXiv preprint arXiv:1811.09556*.

Xu, Kaihe et al. (2015), « Privacy-preserving machine learning algorithms for big data systems », *in*: *2015 IEEE 35th international conference on distributed computing systems*, IEEE, pp. 318–327.

Yee, Kyra, Uthaipon Tantipongpipat, and Shubhanshu Mishra (2021), « Image Cropping on Twitter: Fairness Metrics, their Limitations, and the Importance of Representation, Design, and Agency », *in*: *arXiv preprint arXiv:2105.08667*.

Zhang, Lin et al. (2011), « FSIM: A feature similarity index for image quality assessment », *in*: *IEEE transactions on Image Processing* 20.*8*, pp. 2378–2386.

— (2012), « A comprehensive evaluation of full reference image quality assessment algorithms », *in*: *2012 19th IEEE International Conference on Image Processing*, IEEE, pp. 1477–1480.

DOCTORAT BRETAGNE LOIRE MATHSTIC

Université Bretagne Sud

**Titre :** Modèles d'Apprentissage Automatique pour la Détection Multimodale de Comportements Anormaux

**Mot clés :** Apprentissage profond ; Détection d'anomalies ; Comportements humains

**Résumé :** Nos travaux portent sur le développement de modèles pour la détection de comportements humains anormaux en utilisant plusieurs modalités (*e.g.* audio, vidéo, *etc.*) La détection de comportements anormaux a de nombreuses applications, de la sécurité dans les lieux publics au vote électronique en passant par la santé mentale et les handicaps invisibles.

Nous commençons d'abord par définir le domaine dans lequel s'ancre cette thèse en prenant soin d'en tracer les contours. Puisque nous traitons du comportement humain, nous donnons quelques mises en gardes sur le fait de considérer un comportement comme anormal et les potentielles applications.

Ensuite, nous faisons un tour d'horizon de la littérature qui se rapporte à notre sujet. Nous en identifions des limites que nous avons tenté de repousser pendant cette thèse.

Nos résultats se divisent en trois catégories : 1) Trois nouvelles méthodes pour détection de comportements anormaux. La première méthode modélise les comportements comme des mécanismes temporels, la seconde y ajoute un cadre probabiliste et la troisième se base sur la théorie de l'information et l'entropie de Shannon. 2) Des méthodes de modélisation multimodales afin de pallier au manque dans l'état de l'art. 3) Des méthodes pour la modélisation de représentations du comportement humain.

Nous terminons avec une vue d'ensemble de nos travaux et de nos trouvailles, observant les nouvelles limites du domaine et définissant les perspectives pour de nouveau repousser ces limites.

**Title:** Machine Learning Models for Multimodal Detection of Anomalous Behaviors

**Keywords:** Deep learning; Anomaly detection; Human behaviors

**Abstract:** Our work is focused on the development of new models for the detection of anomalous human behaviors using multiple modalities (*e.g.* audio, video, *etc.*) The field of anomalous human behavior detection has many applications. They range from security in public places to electronic voting and include mental health and invisible disabilities.

We first start by defining the field in which this thesis is rooted and we take care of properly drawing its borders. Since we are studying human behavior, we also give warnings on considering a behavior as anomalous — not abnormal — and the potential resulting applications.

Then, we provide an overview of the literature related to our subject and identify its limits in order to push them.

Our results are split in three categories: 1) Three new methods for detecting anomalous behaviors. The first method models behaviors as temporal mechanisms, the second one adds a probabilistic framework to it and the third one is based on information theory and the Shannon entropy. 2) Methods for multimodal modeling to fill a void in the literature. 3) Methods for modeling representations of human behaviors. All these methods are based on deep neural networks.

We finish with an overview of our works in this thesis and our findings, observing the new limits of the field and from there, we define new perspectives for future works to tackle.