DataScientest • com

Rapport Technique d'évaluation

Détection non supervisée de sons anormaux

Promotion :

Bootcamp Data Scientist Mai-Juillet 2022

Participants :

Sylvain DEBIEU
Geneviève FLEURY
Quentin ROTT

Mentor :

Thomas - DataScientest

Contexte

L'objectif du projet est de développer un modèle de machine learning pour la détection de sons anormaux (ASD). Les applications possibles d'un tel outil concernent notamment le domaine industriel où il est important d'identifier en amont les machines défaillantes pour éviter la casse. Un outil permettant de détecter les sons anormaux émis par une machine permettrait donc de faciliter l'étape de maintenance. Le problème est rendu difficile par le fait que les données de sons anormaux sont rares (et de nature diverse) et ne peuvent donc être que partiellement ou pas du tout utilisées pour l'apprentissage.

La détection de sons anormaux par machine learning fait l'objet de la Task 2 du Challenge DCASE (Detection and Classification of Acoustic Scenes and Events, page web). Nous avons abordé ce projet comme un projet de recherche en machine learning, dans l'esprit du DCASE, et non comme un problème appliqué pour un client industriel.

Du point de vue technique :

Le projet se place dans le domaine du deep learning, où les principaux modèles développés sont de type : réseaux de neurones convolutionnels, autoencodeurs ou encore réseaux de neurones modélisant un mélange de distributions Gaussiennes. Les organisateurs ont mis en place deux systèmes de bases qui procurent des performances raisonnables pour le challenge.

Le premier système de détection des anomalies va modéliser la distribution des échantillons normaux à l'aide d'un auto-encodeur. Le score d'anomalie est calculé par l'erreur de reconstruction du son observée. Pour obtenir des scores faibles pour les sons normaux, l'auto-encodeur est entraîné à minimiser l'erreur sur le jeu d'entraînement ne contenant que des sons normaux. Cette méthode ce base sur l'assomption que l'auto-encodeur ne peut pas reconstruire des sons qui ne sont pas utilisés durant l'entraînement du modèle, c'est-à-dire les sons anormaux inconnus. L'architecture de l'auto-encodeur est constitué d'une couche d'entrée prenant en compte 640 features. Ces features proviennent de cinq fenêtres consécutives d'un mel-spectrogramme utilisant 128 filtres mel et une traille de fenêtre de 64ms avec un saut de fenêtre de 32ms. L'encodeur est ensuite composé de quatre couches dense contenant chacun 128 neurones avec une fonction d'activation relu. La couche « bottleneck » est constituée de 8 neurones. Le décodeur est composé des quatre mêmes couches denses que l'encodeur, ainsi qu'une dernière couche dense composée de 640 neurones pour reconstruire les données d'entrées. Le score d'anomalies est ensuite calculé en prenant la distance L2 entre les données d'entrées et les données de sorties de l'auto-encodeur. Les distances sont calculées sur les données d'entraînements et le 90ème percentile est utilisé comme seuil d'anomalie. Quand le score est plus élevé que le seuil, l'échantillon est considéré anormal.

	ToyCar	ToyTrain	bearing	fan	gearbox	slider	valve
AUC domaine source	91.70%	76.98%	56.95%	78.97%	69.22%	78.81%	52.09%
AUC domaine cible	36.64%	26.36%	59.01%	49.19%	62.83%	49.04%	49.86%
AUC partiel	52.79%	50.56%	51.98%	57.52%	58.72%	56.05%	50.36%

Tableau 1: Moyenne arithmétique des scores AUC selon les différentes machines pour le système de base inlier.

Le second système proposé par les organisateurs est un modèle qui utilise les valeurs aberrantes pour la phase de modélisation, c'est un modèle « outlier exposure-based detector ou OE-based detector ». Ce détecteur utilise MobileNetV2 un réseau neuronal de convolution utilisant des couches de convolutions séparables en profondeur permettant de réduire les coûts computationnels. Ce système de base du challenge a pour tâche d'identifier de quelle section le signal audio a été généré. Il va utiliser comme output la probabilité d'appartenance à chaque section par l'intermédiaire de la fonction softmax. Le score d'anomalie est calculé en prenant le log négatif moyen des probabilités prédites pour la bonne section. Ici il s'agit d'une moyenne car pour chaque échantillon audio, plusieurs images sont extraites et utilisés comme données d'entrées. Pour détecter une anomalie, le 90ème percentile des scores d'anomalies est utilisé comme seuil. Quand le score est plus élevé que le seuil, l'échantillon est considéré anormal.

	ToyCar	ToyTrain	bearing	fan	gearbox	slider	valve
AUC domaine source	61.21 %	60.40 %	63.07 %	71.54 %	70.37 %	69.84 %	68.77 %
AUC domaine cible	52.81 %	46.30 %	61.79 %	51.76 %	59.04 %	48.59 %	60.92 %
AUC partiel	52.46 %	51.59 %	57.89 %	57.56 %	56.53 %	56.49 %	65.27 %

Tableau 2: Moyenne arithmétique des scores AUC selon les différentes machines pour le système de base outlier.

Du point de vue économique :

Dans l'ère de l'industrie 4.0, un nouveau type de maintenance a vu le jour, la maintenance prédictive. Le premier type de maintenance est la corrective. C'est la plus simple et la moins efficace. La défaillance est réparée au moment où elle apparaît et entraîne des temps d'arrêts des chaînes de production. Le second type est la maintenance préventive. Les actions de maintenance sont planifiées selon un intervalle de temps. Elles permettent généralement d'éviter les défaillances, mais elle implique des actions de maintenances inutiles qui conduit à une utilisation inefficace des ressources. La démocratisation de l'intelligence artificielle a aujourd'hui mis en avant la maintenance prédictive. Cette maintenance consiste à détecter des anomalies avant que la panne survienne grâce à des outils de prédiction basés sur des données historisées [Diallo2019]. Selon une étude du cabinet McKinsey, la maintenance prédictive permettra aux entreprises d'économiser 630 milliards d'euros d'ici 2025. Cette économie est permise par l'amélioration du taux de rendement global d'un équipement, d'une réduction des coûts de réparation et d'intervention et d'une amélioration et prédictibilité de la qualité du produit [Vaulpane2018].

Du point de vue scientifique :

La tâche 2 du DCASE consiste à la détection d'anomalies en utilisant uniquement des sons normaux afin de se mettre en condition réelle où on ne peut obtenir suffisamment de sons anormaux en nombre et en type. L'édition du challenge de 2021 ayant toujours pour but la détection des anomalies, se focalise cette fois-ci sur les domain-shifts (changement de domaines). Les changements de domaine sont des différences acoustiques provoquées par les conditions opérationnelles de la machine ou par les bruits de l'environnement. Dans cette édition les domain-shifts étaient similaires dans le jeu d'entraînement et le jeu de test ce qui

impliquait deux assomptions. La première est qu'on aurait identifié tous les domain-shifts possibles et que pour chaque échantillon audio ce domaine était connu. La seconde est que les changements de domaines ne se produisent pas trop fréquemment pour que le modèle s'adapte. Dans des conditions réelles, ces assomptions ne peuvent être satisfaites. Par exemple l'identification des changements de domaines dus aux bruits des autres machines de l'usine semble une tâche impossible. Pour cela, il faudrait prendre le son de chaque machine que l'usine contient, dans tous les états opérationnels qu'elles peuvent prendre et selon des localisations différentes dans le cas de machine mobile. C'est donc dans ce contexte que l'édition 2022 de la tâche 2 du DCASE c'est concentré sur des techniques de généralisation de changements de domaines. Pour cela le jeu de données d'entraînement contient des changements domaines labellisés « source » et très peu de données sur un des changements de domaines labellisés « cible ». Le jeu de test est quant à lui composé à moitié de changements de domaine source, et à moitié de cible. Le but est d'établir un modèle de détection d'anomalies se focalisant sur le son de la machine d'intérêt en s'adaptant aux changements de domaines sans les connaître en avance [Dohi2022].

Objectifs

Principaux objectifs du projet

Dans le challenge DCASE, seuls des sons normaux sont fournis pour l'apprentissage (et quelques sons anormaux pour les tests). Il s'agit donc d'un **problème d'apprentissage non supervisé**. Outre ce contexte général, les conditions du challenge ont quelque peu évolué depuis sa première édition en 2020 et nous avons choisi de travailler *a priori* dans les conditions du challenge 2022 (mais *a posteriori*, nous nous sommes parfois replacés dans les conditions 2020 par souci de simplicité) :

- Dans l'édition 2020, les sons des machines sont superposés aux bruits environnants et ce dans différentes conditions de fonctionnement de la machine et/ou dans différents environnements. Toutefois toutes les données du jeu d'entraînement et du jeu de test sont enregistrés dans la même gamme de conditions. On parle de domaine, ici unique.
- Dans l'édition 2022, les sons sont enregistrés dans deux domaines différents (dits « source » et « cible »). La plupart des sons du jeu d'entraînement sont dans le domaine source tandis que les sons du jeu de test (du moins celui que nous avons utilisé[1]) sont à 50/50 dans les domaines source et cible. On parle de **« domain shift ».**

Précisons que lorsque nous avons commencé ce projet, le challenge DCASE était encore ouvert. A l'époque, seules les deux méthodes de benchmark introduites ci-dessus étaient présentées. Les résultats précisant les scores obtenus par les différentes équipes ainsi qu'un résumé des méthodes utilisées ont été publiés début juillet.

Contact avec des experts métier

- Aucun de nous trois n'est entrés en contact avec un expert métier.

[1] A noter que dans les conditions réelles du challenge DCASE 2022, les modèles proposés sont évalués par les organisateurs du challenge sur un jeu d'évaluation (que nous n'avons pas utilisé) et dans lequel le label du domaine d'appartenance d'un son (domaine source ou cible) n'est pas spécifié.

Niveau d'expertise préalable des membres de l'équipe

- **Sylvain Debieu :**
  - Connaissances préalables sur la transformée de Fourier, spectrogrammes (chimie analytique et résonnance magnétique nucléaire)
  - Aucune expérience en programmation
  - Aucune connaissance en data science autre que de la culture général (AlphaZero,GPT-3, AlphaFold)

- **Geneviève Fleury :**
  - Connaissances préalables sur les ondes sonores, transformée de Fourier, spectrogrammes (physicienne de métier)
  - Expérience en programmation python
  - Aucune connaissance en data science, machine learning, deep learning au début de la formation

- **Quentin Rott :**
  - Connaissances préalables sur la transformée de Fourier (Cristallographie à rayons X)
  - Expérience en programmation: Python, SQL, C#
  - Connaissances préalables en machine learning (Autodidacte). Aucune connaissance en deep learning au début de la formation.
  - Aucune connaissance préalable dans la détection d'anomalies

Data

Description du jeu de données

Structure et volumétrie du jeu de données

Nous avons utilisé une partie du jeu de données du challenge DCASE 2022 – Task 2 mis librement à disposition sur la page web du challenge. En l'occurrence, nous avons utilisé le jeu de données nommé « development data set » dont la structure est représenté sur la Figure 1. Le jeu de données est constitué de **25200 clips sonores de 10 secondes** provenant de l'enregistrement du bruit émis par **7 types de machines** ('fan', 'gearbox', 'bearing', 'slider', 'toy car', 'toy train', 'valve') **mixé avec des bruits environnementaux provenant d'usines**. Les machines sont uniformément représentées (3600 clips par machine) et le jeu de données s'organise ainsi :

- Pour chaque type de machine, le jeu de données est divisé en :
  - **un jeu d'entraînement** de 3000 clips tous normaux
  - **un jeu de test** de 600 clips dont la moitié correspond à des sons normaux et l'autre moitié à des sons anormaux (obtenus en détériorant volontairement les machines)
- Pour chaque type de machine, que ce soit dans le jeu d'entrainement ou de test, les clips sonores sont répartis uniformément en **3 sections** de 1000 (train set) + 200 (test set) clips. Chaque section correspond à une variable qui a été modifiée d'un enregistrement à un autre. Par exemple, pour les boites de vitesse ('gearbox') :

- la section 0 regroupe 1200 sons enregistrés pour différents voltages appliqués aux boites
- le section 1 regroupe 1200 sons enregistrés pour différentes charges appliquées aux boites
- la section 2 regroupent 1200 sons enregistrés à partir de différentes boites

- Pour chaque section de chaque type machine, les sons sont regroupés par **attribut** correspondant à un régime de fonctionnement de la machine et/ou aux conditions caractérisant l'environnement. Pour l'exemple de la boîte de vitesse, la section 0 est divisée en 13 sous-groupes de clips sonores indexés par la valeur (dit attribut) du voltage V appliqué aux boites (V=1 volt, V=2 volt etc …). Cette répartition n'est pas uniforme. Certains attributs sont beaucoup plus représentés que d'autres. Ceci est illustré sur la figure 2 dans l'exemple de la 'gearbox'. En particulier, les attributs sont regroupés en 2 catégories disjointes dits domaines :
  - Le **domaine source** (par exemple V=1 volt, V=2 volt, …) qui représente 99% du jeu d'entrainement et 50% du jeu de test
  - Le **domaine cible** (par exemple V=1.3 volt, V=2.8 volt, …) qui représente 1% du jeu d'entrainement et 50% du jeu de test. L'emploi du mot cible ne fait pas référence à une variable cible ici.

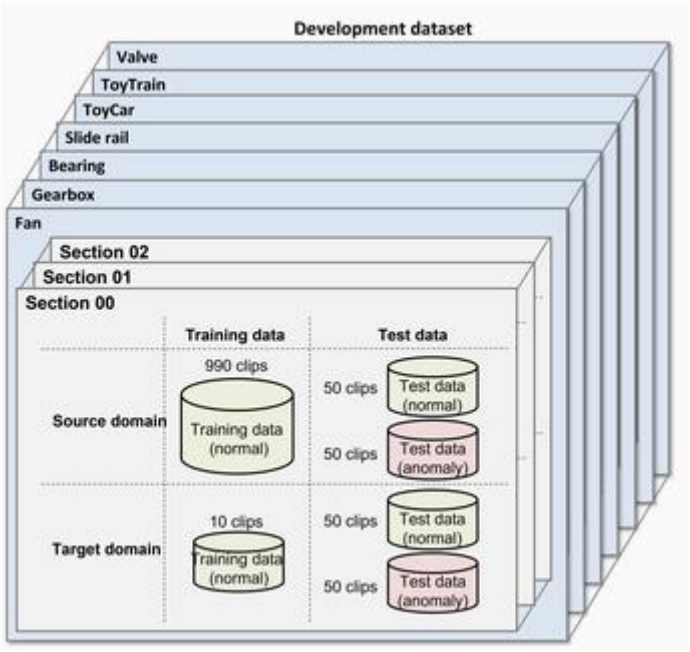


Figure 1 - Structure du jeu de données considéré. Image issue de la page web du challenge DCASE 2022.

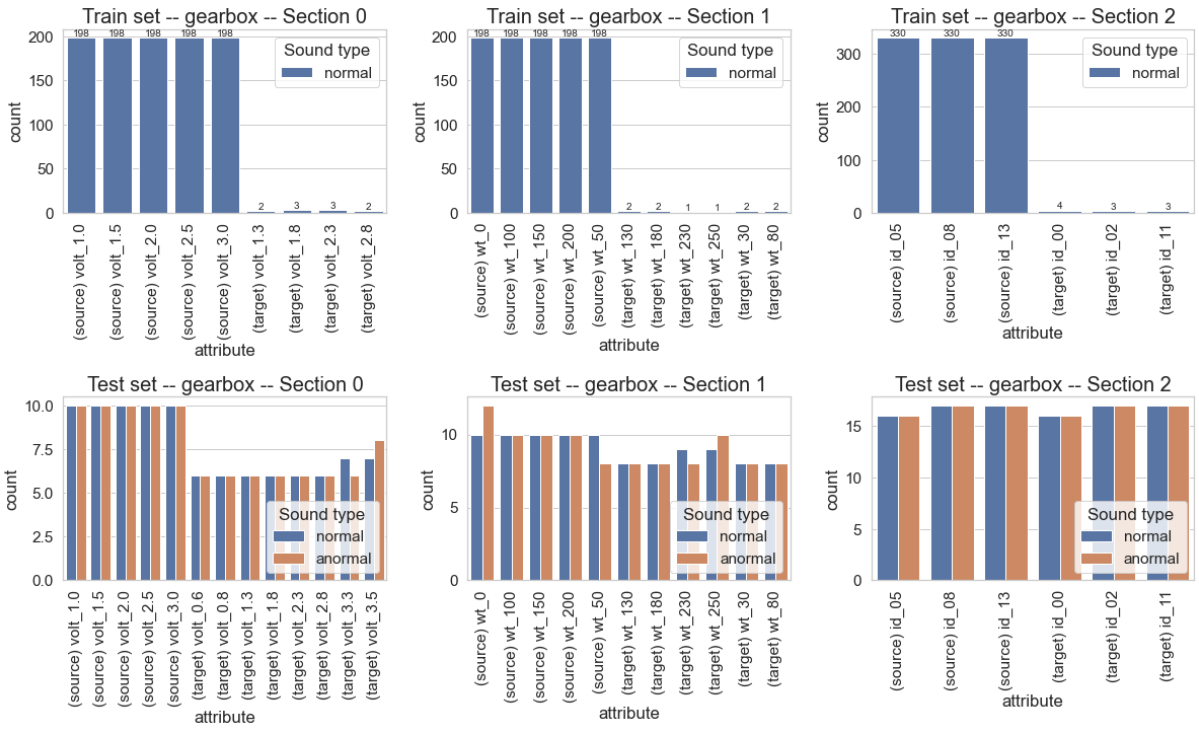


Figure 2 - Répartition des attributs par section dans les jeux d'entrainement (en haut) et de test (en bas) correspondant aux boites de vitesses ('gearbox'). Les couleurs bleue et rouge des barres correspondent respectivement au caractère normal et anormal des sons.

Identification des variables explicatives du jeu de données

Chacun des 25200 éléments du jeu de données se compose :

- **de 6 variables qualitatives** (comme vu au-dessus):
  - le type de jeu de données (train ou test set)
  - le type de machine (parmi 7)
  - la section (0, 1 ou 2)
  - l'attribut (variant selon les machines)
  - le type de domaine (source ou cible)
  - la nature du son (normal ou anormal, tous les sons du train set étant normaux)
- **d'un fichier sonore au format wav d'une durée de $Dt = 10$ secondes**. Chaque son a été enregistré avec un microphone fixe mono-canal relevant la valeur de la pression acoustique $P(x,t)$ au point fixe x et au temps t, c'est-à-dire la variation (due à l'onde sonore) de la pression locale par rapport à la pression atmosphérique. Ce signal analogique a ensuite été discrétisé avec une fréquence d'échantillonnage $fe = 16$ kHz en un signal numérique de $Dt \times fe = 160000$ points. Un fichier sonore brut de notre jeu de données est donc constitué de **160000 variables quantitatives réelles**, correspondant à la discrétisation $P_i(x,t_i)$ de la pression acoustique au temps t_i (notée $P_i(t_i)$ dans la suite)

Remarque sur l'existence de jeux de données additionnels

Nous nous sommes limités pendant ce projet à ce jeu de données déjà conséquent mais d'autres données similaires auraient pu également être utilisées :

- le jeu d'entraînement additionnel (« additional training dataset ») du challenge DCASE 2022 – Task 2
- les données des précédentes éditions du challenge (DCASE 2020 et DCASE 2021) issues respectivement des data sets ToyADMOS & MIMII et ToyADMOS2 & MIMII DUE

- les données des data sets AudioSet, IDMT-ISA-ELECTRIC-ENGINE, OpenL3, PANNs, PyTorch Image Models, VGGish, torchvision.models, disponibles sur la page du Challenge DCASE 2022.

Pre-processing et analyse des données

Pre-processing des fichiers audio

Pour traiter et analyser les fichiers audio, nous avons utilisé le package python librosa (voir l'excellent cours de Valerio Velardo [Velardo2020]). Nous avons commencé par explorer différentes manières de représenter numériquement un son du jeu de données :

- par la **série temporelle** $P_i(t_i)$, c'est-à-dire en affichant les données brutes, sans pré-traitement
- par le **spectre de Fourier** $\left|\widetilde{P}_i(f_i)\right|$, correspondant à l'amplitude de la transformée de Fourier discrète (DFT) de la série temporelle $P_i(t_i)$. Ceci est très couteux numériquement.
- par un **spectrogramme** en représentation mixte temps (s) – fréquence (Hz). Celui-ci est obtenu en découpant la série temporelle en plusieurs fenêtres et en effectuant une DFT sur chacune d'entre elles pour en extraire l'**amplitude** $\left|\widetilde{P}_i(t_i, f_i)\right|$. Deux paramètres sont particulièrement essentiels ici : « n_fft » et « hop_length » contrôlant la résolution temporelle et le chevauchement des fenêtres. Par exemple, plus le nombre de points « n_fft » pris pour chaque DFT est petit, plus la résolution temporelle est bonne mais à l'inverse moins la résolution fréquentielle est bonne.
- par un **mel-spectrogramme**, également en représentation mixte temps-fréquence. Dans ce cas, l'axe des fréquences est exprimée en échelle de mel, plus adaptée à la perception humaine des fréquences.
- par un **(mel-)spectrogramme de la phase** de la DFT et non de l'amplitude (la phase donnant une information sur le sens de la flèche du temps, contrairement à l'amplitude).
- par une **transformée en ondelettes** de la série temporelle $P_i(t_i)$en utilisant la librairie PyWavelets (voir lien du guide https://ataspinar.com/) [PyWav]. Basée sur le même principe que la transformée de Fourrier, elle offre une haute résolution fréquentielle mais une faible résolution temporelle pour les petites valeurs de fréquence et une forte résolution temporelle mais une faible résolution fréquentielle pour les grandes valeurs de fréquence .

La figure 3 résume les principales manières de visualiser des données audio.

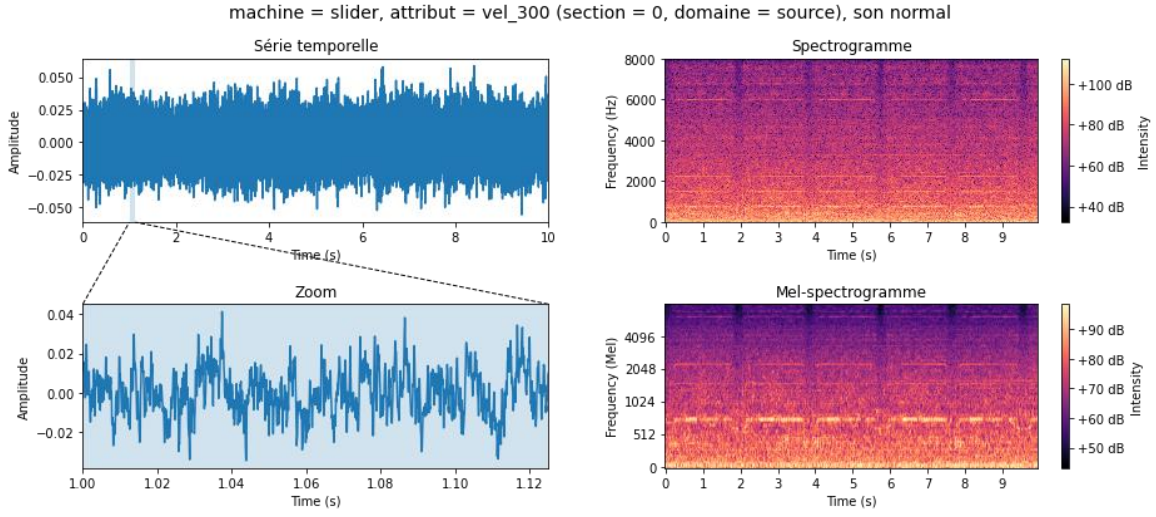


Figure 3 - Exemples de représentations numériques typiques d'un fichier audio (d'autres sont possibles, voir texte). Ici le son provient d'une glissière ('slider'), est normal, et est caractérisé par l'attribut 'vel=300' correspondant à sa vitesse de fonctionnement. A gauche : amplitude de la pression acoustique en fonction du temps. A droite :

spectrogramme (en haut) et mel-spectrogramme (en bas) après transformée de Fourier discrète. Les deux sont en échelle linéaire sur l'axe des fréquences. Le code couleur représente l'intensité sonore en décibels (dB).

Analyse des fichiers audio et des spectrogrammes

Nous avons ensuite analysé un échantillons de sons, à la fois à l'oreille (en écoutant des fichiers wav) et à l'œil en examinant les (mel-)spectrogrammes correspondants. Nous avons joué en particulier sur les paramètres « n_fft » et « hop_length » ainsi que sur l'échelle (linéaire ou log) de l'axe des fréquences des spectrogrammes.

Nous avons constaté d'une part que les sons normaux peuvent considérablement varier même lorsqu'ils proviennent d'un même type de machine partageant le même attribut. Parfois, la différence entre 2 sons ayant le même attribut est apparemment plus important qu'entre 2 sons ayant des attributs différents (mais provenant du même type de machine). Ceci est illustré sur la figure 4 et est dû (sans doute) au fait que les enregistrements des sons ont été réalisés en présence de l'environnement des machines. Autrement dit, chaque clip sonore est la « superposition » de sons provenant de la machine (et caractérisé par l'attribut) et de sons provenant de l'environnement (parfois caractérisé par l'attribut mais pas toujours). Il reste ainsi une certaine ambiguïté quant à la caractérisation de chacun des sons et ceci complique bien sûr notre problème.

De la même manière, nous avons comparé des couples de sons normaux et anormaux, en particulier lorsqu'ils proviennent d'un même type de machine et partagent le même attribut. Quelques exemples de spectrogrammes sont montrés sur la Figure 5. Dans certains cas, les sons anormaux sont clairement distinguables à l'oreille et leur signature est visible sur le spectrogramme. Il est remarquable toutefois que l'anomalie perceptible à l'oreille ne correspond parfois qu'à une zone très réduite du spectrogramme et en particulier à une variation très petite par rapport aux variations dues au changement d'échantillon (voir la 2ème ligne de la figure 5). Dans d'autres cas en revanche, il ne nous pas été possible de distinguer l'anomalie, que ce soit à l'oreille ou en examinant le spectrogramme. Il n'est pas exclu malgré tout qu'une oreille experte de telle ou telle type de machine ait été capable de le faire. Il est aussi envisageable que certaines anomalies correspondent à un régime de fréquence inaudibles par l'oreille humaine.

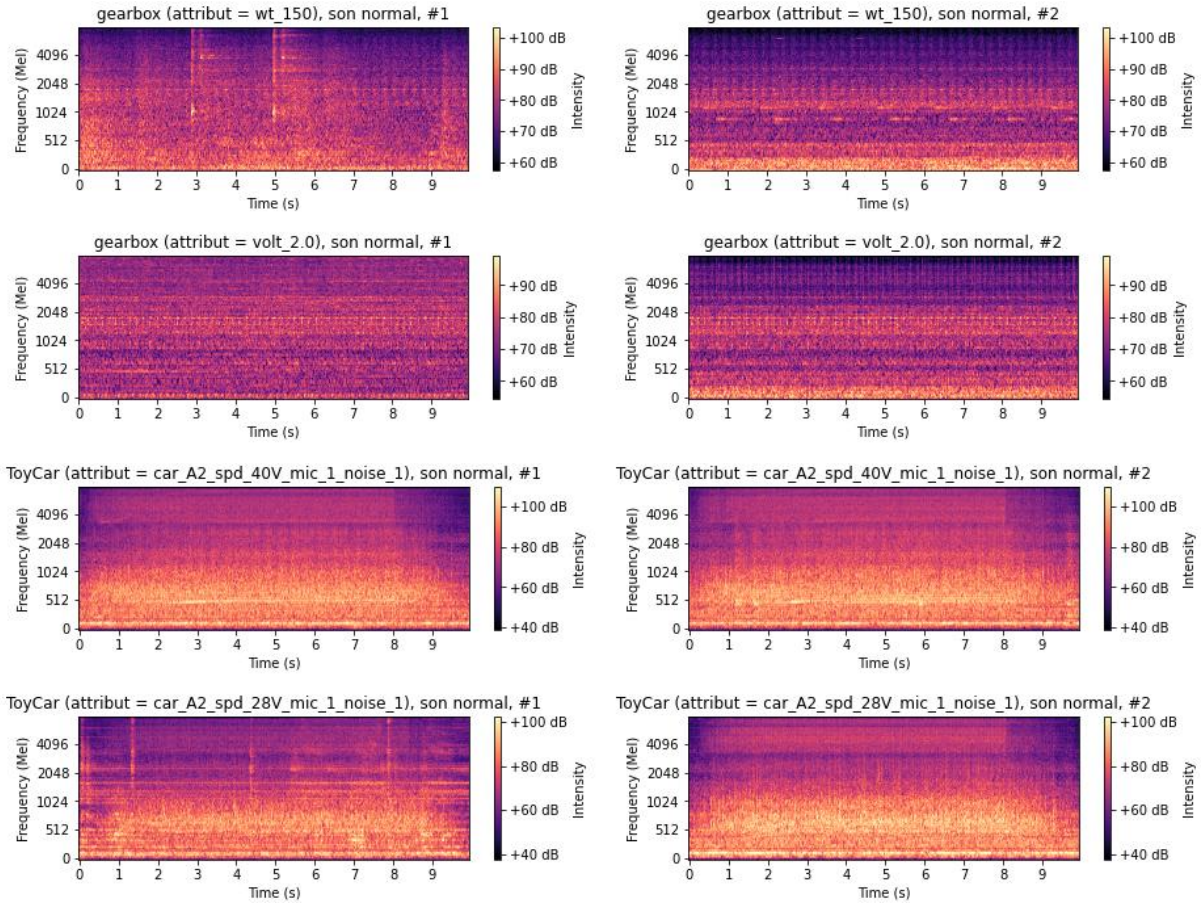


Figure 4 - Exemples de mel-spectrogrammes de sons normaux issus du jeu d'entraînement. Sur une même ligne sont comparés deux sons provenant du même type de machine et partageant le même attribut : des différences qualitatives entre les 2 spectrogrammes sont clairement visibles dans certains cas, tandis que dans d'autres cas, les spectrogrammes se ressemblent.

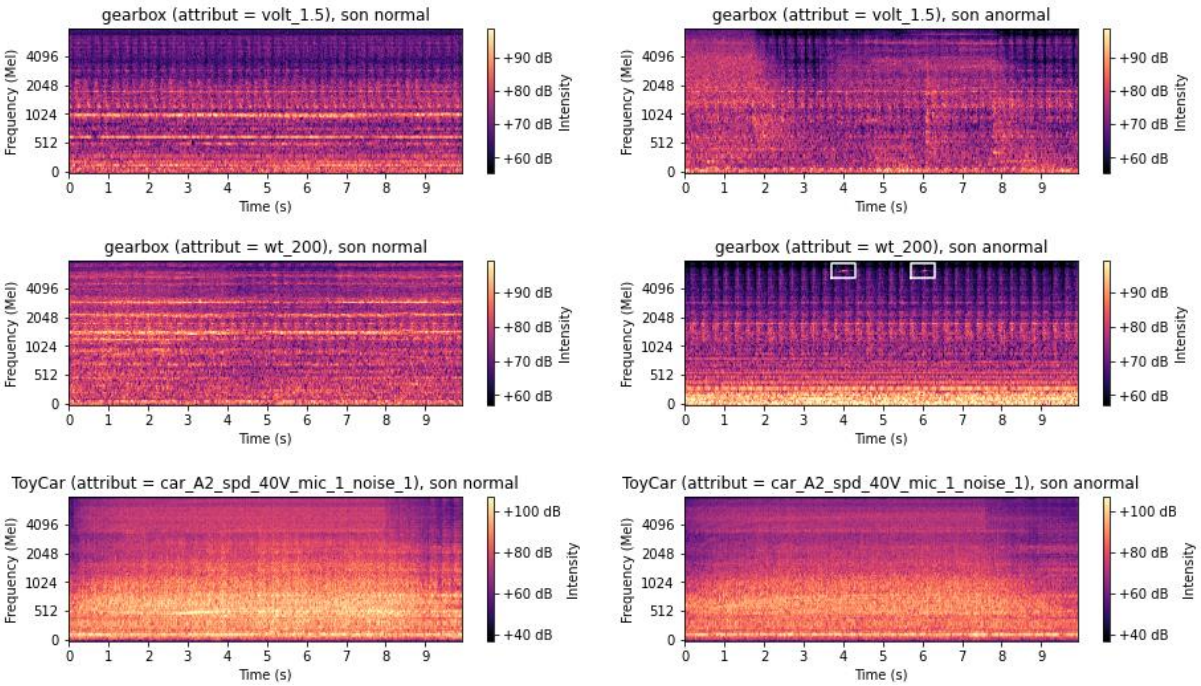


Figure 5 – Mel-spectrogrammes de sons normaux (à gauche) et anormaux (à droite). Les lignes correspondent à différentes machines et/ou attributs. La comparaison (par ligne) est faite entre 2 sons provenant du même type de machine et ayant le même attribut. L'anomalie est visible sur les mel-spectrogrammes des deux premiers sons (fond diffus pour le premier et couinements brefs à haute fréquence pour le second (encadrés en blanc) correspondant à ce qui s'entend clairement à l'oreille). En revanche, l'anomalie n'est pas clairement identifiable sur le mel-spectrogramme du bas (et est aussi inaudible pour une oreille non experte).

Principales difficultés identifiées après l'étape d'analyse du jeu de données

- Le problème est **non-supervisé** : nous cherchons *in fine* à prédire le caractère normal ou anormal d'un son mais tous les sons du jeu d'entraînement sont normaux.
- Il existe un **« domain shift »** dans les données : 99% des sons du jeu d'entraînement sont dans le domaine « source » (et 1% dans le domaine « cible ») tandis que dans le jeu de test, la répartition des données entre domaines « source » et « cible » est équilibré (50/50).
- Il y a de **multiples façons de pré-traiter les données** : série temporelles, spectrogrammes (en échelle linéaire ou log, en mel ou Hz, choix des paramètres n_fft et hop_length, …), transformées en ondelettes, éventuels filtres haute/basse fréquence, …
- **Les anomalies ne sont pas toujours clairement identifiables** sur les spectrogrammes (ou à l'oreille), d'autant plus que de grandes variations peuvent aussi apparaître entre 2 sons normaux de même attribut. Ainsi il est « parfois » (à quantifier) difficile de distinguer les variations dues à une anomalie et celles dues à la statistique.
- **Une image de spectrogramme diffère qualitativement d'une image standard** obtenue par un capteur optique [Rothmann2018] : les axes (temps-fréquence) représentent deux quantités différentes, les propriétés spectrales acoustiques sont non-locales, etc ... Ainsi, en particulier, les techniques usuelles d'augmentation de données utilisées pour des images n'ont pas *a priori* de sens pour des images de spectrogrammes.

Projet

Classification du problème

Le problème de détection d'anomalies peut être traité par des approches supervisées ou non supervisées. Une approche supervisée est la classification binaire permettant de prédire si un échantillon est normal ou anormal. Pour les approches non supervisées plusieurs techniques peuvent être appliquées, les techniques de plus proches voisins, les méthodes basées sur les distances ou encore ceux basées sur la densité.

Nous avons d'abord débuté une première approche avec une classification binaire à l'encontre des règles du défi qui ne permettent pas d'utiliser les échantillons anormaux pour la modélisation. Nous sommes ensuite passés à des approches respectant cette fois ci les conditions du challenge : nous avons utilisé des méthodes basées sur les distances ou sur les probabilités de tâches annexes de classification, avec la supposition que les anomalies sont écartées de la majorité des autres points.

La tâche de machine learning du projet s'apparente à la détection d'anomalies.

La métrique AUC ROC (Area Under the Curve de la courbe Receiver Operating Characteristic) a été utilisée pour comparer nos modèles.

Dans certains cas, nous avons également analysé la précision (« accuracy ») de nos modèles car il s'agit d'une métrique plus facilement interprétable. De façon générale, nous avons affiché les tables de confusion et rapports de classification de nos modèles

Choix du modèle & Optimisation

Classification supervisée par machine learning

Ici le label normal/anormal des sons est connu pour l'entraînement et il s'agit de prédire ce label sur un jeu de test. Plusieurs approches par machine learning ont été suivies :

- En utilisant les spectrogrammes d'amplitude comme données d'entrée, machine par machine, nous avons testé plusieurs méthodes de réduction de dimensions (moyenne sur l'axe des fréquences, aplatissement du spectrogramme sur un vecteur puis sélection de features avec SelectPercentile et/ou PCA), ainsi que plusieurs méthodes de classification (xgboost, random forest, svm, KNN) en jouant sur les hyper-paramètres. Nous avons aussi essayé d'utiliser les spectrogrammes d'amplitude et de phase simultanément ce qui a conduit à une légère amélioration des performances (AUC) mais négligeable devant l'augmentation du temps de calcul et nous avons abandonné cette idée. Les meilleurs résultats ont été obtenus en faisant de la PCA (réduisant le nombre de features à environ 550) puis du xgboost. Ils sont illustrés sur la figure 6 et montrent un $AUC \gtrsim 0.8$ et un taux d'erreur de classification $\eta \lesssim 25\%$ pour toutes les machines (et jusqu'à $AUC = 0.98$, $\eta = 7\%$ pour le « bearing »).
- En utilisant les coefficients donnés par l'application des ondelettes de Daubechies, des attributs (entropies, cross-zéros,...) ont été générés pour obtenir au final environ 170 features pour chaque fichiers audio, qui ont ensuite été utilisées dans un classifier type Gradient Boosting pour déterminer si le son était anormal ou non. L' AUC varie entre 0.65 et 0.89 suivant le type de machines, sachant que cela aurait pu être optimisé en faisant varier les audios normaux. En effet, le peu d'échantillons anormaux du dataset nous ont poussé à limiter le nombre de sons normaux pour éviter un déséquilibre lors de l'apprentissage.

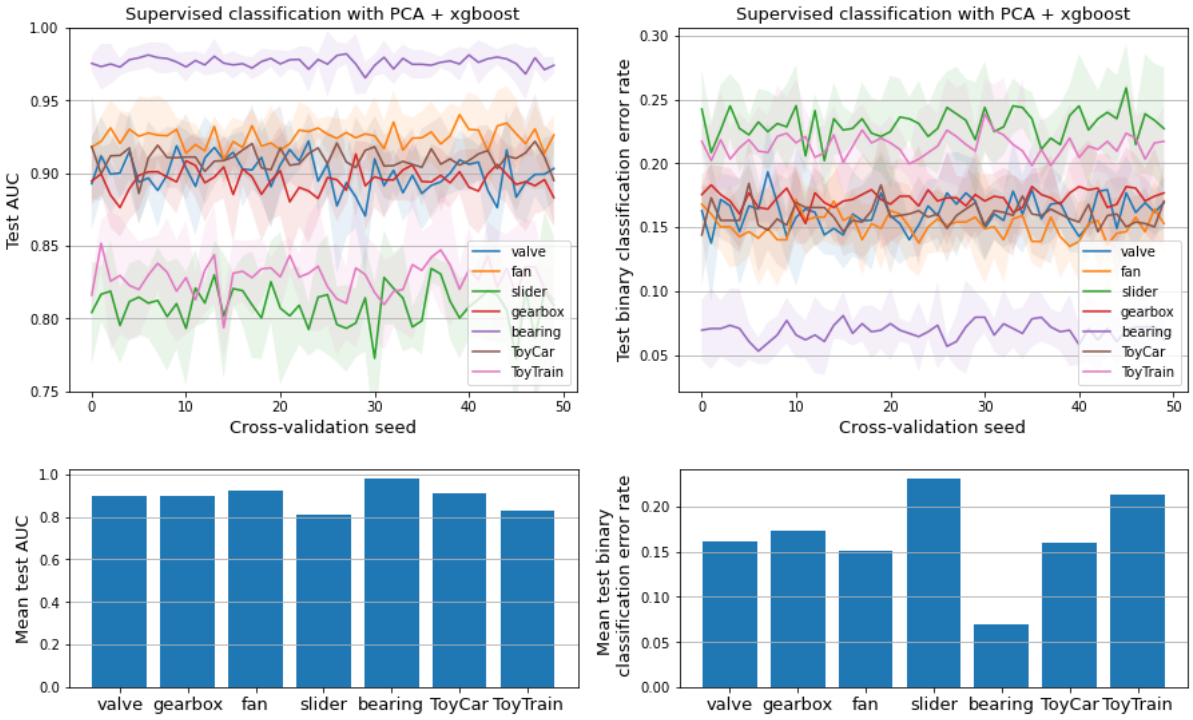


*Figure 6 - Résultats obtenus avec un **modèle de classification supervisée** (labels normal/anormal des sons connus). Le modèle a été ajusté sur la 'gearbox' en utilisant comme inputs les spectrogrammes (d'amplitude) des sons, aplatis. Le modèle retenu est un modèle avec réduction de dimensions (SelectPercentile(30%) + PCA(85%)) et du gradient boosting (xgboost + optimisation des hyperparamètres). Ici (seulement) 500 sons normaux et 300 anormaux ont été utilisés par machine (pour limiter le temps de calcul). La validation croisée a été réalisée sur 5 feuillets.*

Classification supervisée par deep learning

Une méthode utilisant des couches de convolution 1D n'ayant pas donné de résultats valables, une autre méthode utilisant des couches de convolution 2D fut appliqués sur les données générés par l'utilisation d'une transformée de Fourier locale sur les données audio. Dans un premier temps, ce modèle fut utilisé sur une seule machine pour déterminer si une anomalie était présente. Une généralisation du modèle permet de classifier par type de machines avec une grande précision. Une classe en plus permet de regrouper les anomalies pour tout type de machine. Les résultats sont résumés ci-dessous (Figure 7):

	precision	recall	f1-score	support
0	0.78	0.63	0.69	417
1	0.70	0.82	0.76	133
2	0.96	0.98	0.97	109
3	0.76	0.82	0.78	114
4	0.96	0.97	0.96	127
5	0.74	0.96	0.84	111
6	0.87	0.91	0.89	129
7	0.86	0.91	0.88	120
accuracy			0.82	1260
macro avg	0.83	0.87	0.85	1260
weighted avg	0.82	0.82	0.81	1260

col_0	0	1	2	3	4	5	6	7
row_0								
0	261	46	4	29	4	38	18	17
1	24	109	0	0	0	0	0	0
2	2	0	107	0	0	0	0	0
3	19	0	0	93	1	0	0	1
4	3	0	0	1	123	0	0	0
5	4	0	0	0	0	107	0	0
6	11	0	0	0	0	0	118	0
7	11	0	0	0	0	0	0	109

Figure 7 – Tableaux reprenant les résultats obtenus avec un modèle DL de convolution 2D entrainé sur tous type de machines. La classe 0 représente les anomalies détectées sans distinctions de machines. A gauche, le rapport de classification de la méthode. A droite, le tableau croisée entre les vrais attributions et celles prédites par le modèle.

Classification non-supervisée par deep learning

a) <u>Classification des sons après une première tâche de classification supervisée du type de machine par des méthodes de Deep Learning basiques</u>

Nous avons entraîné quelques modèles de Deep Learning très simples (typiquement 1 à 3 couches de CNN et 1 à 3 couches Denses avec/sans « Batch Normalization » et « Drop out ») à reconnaître le type de machine. D'autres modèles plus évolués et optimisés pour cette tâche sont discutés ci-dessous. L'idée dans un premier temps était de mettre en place une méthodologie permettant de prédire le caractère normal/anormal d'un son à partir des probabilités p_m (calculées par le modèle), p_m étant la probabilité que le son provienne de la machine m. Nous avons fait l'hypothèse que

$$p_{anormal} \equiv 1 - \max_m p_m$$

peut s'interpréter comme la probabilité qu'un son soit anormal. A partir de cette probabilité, nous avons pu calculé un AUC par machine. Pour prédire le label normal/anormal d'un son (0 ou 1), nous avons également introduit la notion de seuil de probabilité p_c : si $p_{anormal} > p_c$, le son est classé comme anormal et inversement. Nous avons alors analysé comment la précision binaire (associée au caractère normal/anormal) de notre approche dépend du seuil p_c. En pratique, la classification des machines est réalisée par l'algorithme avec une précision proche de 1 et les p_m sont toutes très proches de 1, si bien que $p_{anormal} \approx 0$. Ainsi le choix de p_c à l'œil à partir des courbes de la précision *vs* p_c n'est pas aisé. Pour contourner ce problème, nous avons défini un score de précision

$$anomaly\ score = g^{-1}(p_{anormal})$$

où g est la fonction sigmoïde (en tronquant les bornes). D'autres définitions auraient pu être choisies. Un son est alors étiqueté comme anormal si son score d'anomalie est plus grand qu'un score critique. Nos résultats consignés sur la figure 8 montrent qu'en ajustant le choix de ce score critique, les sons provenant du slider peuvent être étiquetés correctement avec une précision de l'ordre de 75%. Les sons des autres machines en revanche sont beaucoup moins bien étiquetés.

La même approche a été reproduite en considérant uniquement les données du domaine source (pour simplifier le problème, conditions du challenge 2020) et en séparant les données des jeux de test section par section (le label « section » étant connu dans les deux jeux, sans outrepasser les règles du challenge). Nos résultats sont consignés sur la figure 9 et montrent une grande variation de l'AUC d'une machine d'une section à l'autre. Par l'exemple l'AUC du ToyCar s'approche des 90% dans la section 2 mais tombe à 50% environ dans les sections 0 et 1. L'AUC du slider est toutefois constant (environ 80%) pour toutes les sections.

De façon générale, nous constatons qu'un modèle simple de Deep Learning suffit pour classifier le type de machine avec une précision moyenne >97% (le label du type de machine étant connu). En revanche, déduire un label normal/anormal à partir de cette tâche auxiliaire n'est pas trivial. L'approche testée ici n'est pas satisfaisante car (comme le montre la figure 8 en haut à gauche) il n'existe pas de score critique au-dessus [en-dessous] duquel la plupart des sons seraient anormaux [normaux].

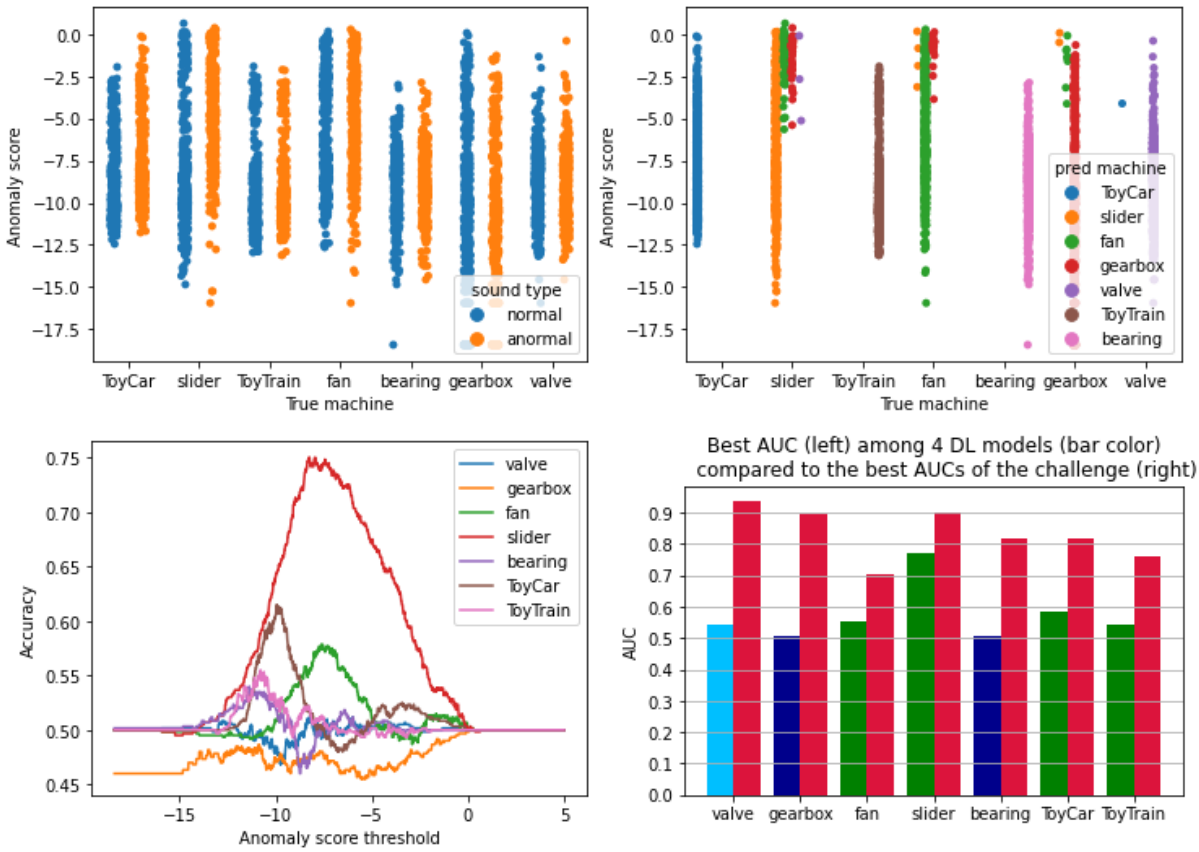


*Figure 8 - **Classification non supervisée de sons normaux/anormaux avec une approche Deep Learning** : un modèle a été entraîné à reconnaitre le type de machine sur tout le jeu d'entraînement (20% ayant été conservé pour la validation). Plusieurs modèles similaires à base de couches convolutionnelles et couches denses ont été testés en prenant les mel-spectrogrammes 2D comme données d'entrée. Puis pour chaque son du jeu de test a été calculé un score d'anomalie (relié à la probabilité de détection du type de machine, voir texte) : plus ce score est élevé, plus la probabilité que le son soit anormal devrait être élevé. **Figures du haut :** les sons provenant de machines non reconnues ont un score d'anomalie élevée (à droite) mais (à gauche) il n'existe pas à l'oeil de seuil au-dessus [en-dessous] duquel la majorité des sons seraient anormaux [normaux]. **Figures du bas :** La précision ('accuracy') de prédiction des sons normaux/anormaux dépend fortement du type de machine et du seuil pris pour étiqueter les sons (à gauche). De même, l'AUC dépend fortement du type de machine et du modèle employé (à droite ; par machine seul l'AUC du meilleur des 4 modèles testés avec cette approche est montré et comparé avec le meilleur AUC obtenu par l'équipe gagnante du challenge).*

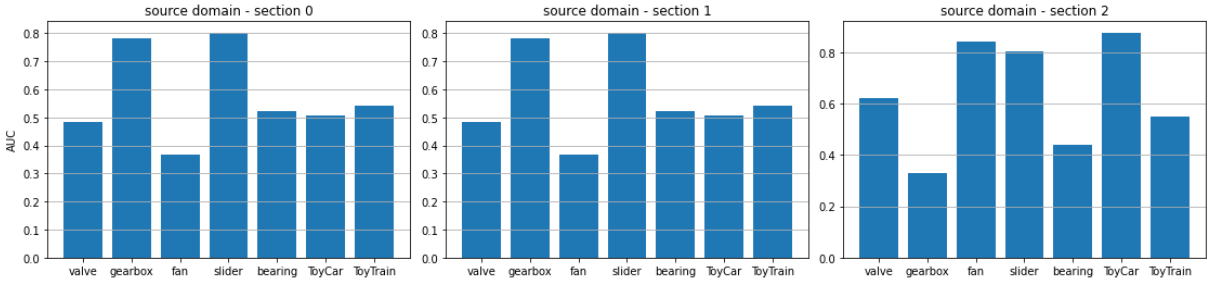


*Figure 9 - **Classification non supervisée de sons normaux/anormaux avec une approche Deep Learning** : même approche que celle de la figure 8 en considérant uniquement les données du domaine source et en faisant l'entraînement et la prédiction séparément pour chaque section. Ici seul un modèle a été testé. Pour le « slider », l'AUC associé à la reconnaissance du caractère normal/anormal des sons est « bon » quelque soit la section. A l'inverse pour la « gearbox » ou le « fan », l'AUC dépend fortement de la section.*

b) ***Approche d'embedding avec FaceNet puis classifieur***

Nous avons testé une autre approche combinant les algorithmes FaceNet et forêts aléatoires. Dans un premier temps, les mel-spectrogrammes des sons (de taille 313x128) sont transformés en vecteurs d'embedding de taille 128 avec la méthode FaceNet + triplet loss. Nous avons effectué cette étape d'embedding soit en regroupant par section, soit en

regroupant par type de machine. Pour évaluer cette tâche, nous avons effectué une analyse en PCA (voir figure 10) en espérant observer des groupes séparés entre sections ou machines (ce qui est globalement le cas) et également des groupes bien séparés entre sons normaux et anormaux (connaissant le label, utilisé pour l'analyse PCA mais pas pour l'embedding). Bien que nous observions une distribution non homogène entre sons normaux et normaux, il n'y a pas de frontière claire entre les 2 groupes (la meilleure séparation est illustrée sur la figure 10). Ceci nous a dissuadé d'utiliser dans un second temps une approche de clustering (mais ceci pourrait être testé) et nous avons complété la méthode par un algorithme de forêts aléatoires prédisant la section ou le type de machine. Nous avons alors suivi la méthodologie discutée ci-dessus pour prédire le caractère normal/anormal d'un son et en déduire la précision et l'AUC. Nos résultats (voir Fig.10) sont légèrement meilleurs que ceux de la figure 8 pour la valve et la gearbox mais légèrement moins bons pour le ToyCar.

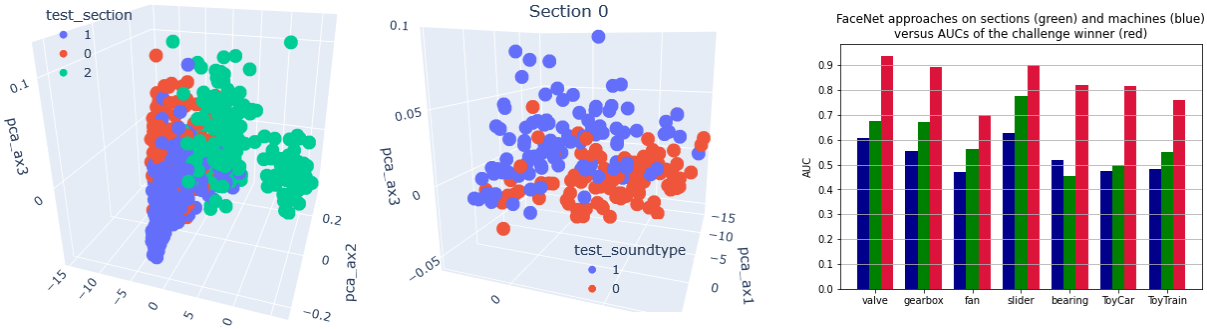


Figure 10 - ***Classification non supervisée de sons normaux/anormaux avec une approche FaceNet + Random Forest :*** *les vecteurs d'embedding sont construits à partir des mel-spectrogrammes en tentant de rapprocher les images par section. L'analyse en PCA montre (ici pour le slider) une bonne séparation des sons provenant de sections différentes (à gauche) et une séparation correcte (du moins visible) des sons normaux et anormaux (au milieu). Pour les autres machines, la séparation n'est pas aussi nette, voire indétectable à l'œil. A droite : AUCs obtenus en appliquant l'approche FaceNet sur les sections (en vert) et sur le type de machine (en bleu), comparés aux AUCs obtenus par les gagnants du challenge (en rouge).*

c) ***Approches par auto-encodeurs (Sylvain)***

Des modèles d'autoencoders ont été développés avec de multiples approches:

En utilisant les transformée de Fourier locale en inputs et outputs, en faisant entrer les valeurs logarithmiques de celles-ci, avec différentes tailles de bottleneck (varier entre 5 et 40 units), mais cela n'a pas donné de résultats exploitables.

En utilisant les suffixes des sons, qui déterminent le bottleneck et qui seront ensuite utiliser pour générer une partie décodeur chacun. Pour caractériser le son, il passe donc par l'encodage (qui fonctionne très bien), pour prédire son suffixe, il passe ensuite dans les différents décodeurs qui génèrent chacun une erreur quadratique moyenne qui sont ensuite additionnés. Malheureusement, les différences en sortie ne sont pas significatives et ne permettent pas de déterminer la normalité du son.

Une autre méthode explorée fut l'utilisation du découpage du sons d'origine en plusieurs sons. Ensuite , à l'aide des suffixes en bottleneck, l'algorithme recompose les 5 parties du sons à partir du son original. Encore une fois , les résultats ne sont pas pertinents pour une utilisation future.

d) ***Approches par auto-encodeurs (Quentin)***

d-1) Tâche auxiliaire de classification des types de machines:

Notebook : Iteration 2 Searching for best CNN for classification task.ipynb

Une première itération a consisté à créer un classifieur qui avait pour tâche d'identifier le type de machine. Pour cela un réseau neuronal convolutif a été modélisé. Le but de cette première itération était d'extraire des features spécifiques à chaque type de machines afin d'utiliser les outputs des couches intermédiaires du CNN en tant que données d'entrées pour des modèles d'auto-encodeurs. Cette tâche auxiliaire d'apprentissage des métadonnées est inspirée par les travaux effectués par Chen & al. [Chen2022].

La modélisation a débuté par la recherche des architectures convolutives les plus prometteuses. Pour cela deux types d'architectures ont été analysées. La première fut un modèle répétant une couche de convolution avec une couche de pooling et la seconde consistait à la répétition de deux couches de convolutions avec une couche de pooling. À chaque répétition le nombre de filtres de convolutions étaient multipliés par deux. La taille des kernels resta constante et fut initialisée à 3. La couche de pooling utilisait une taille de kernel de 2 et un stride de 2. Les modèles se terminaient par une couche d'aplatissement suivie d'une couche dense de 128 neurones, et enfin une dernière couche dense de 7 neurones correspondant aux 7 types de machine. Les modèles ont été entraînés en utilisant les images de mel-spectrogrammes de format 128 filtres mel x 313 frames du jeu de données d'entraînements.

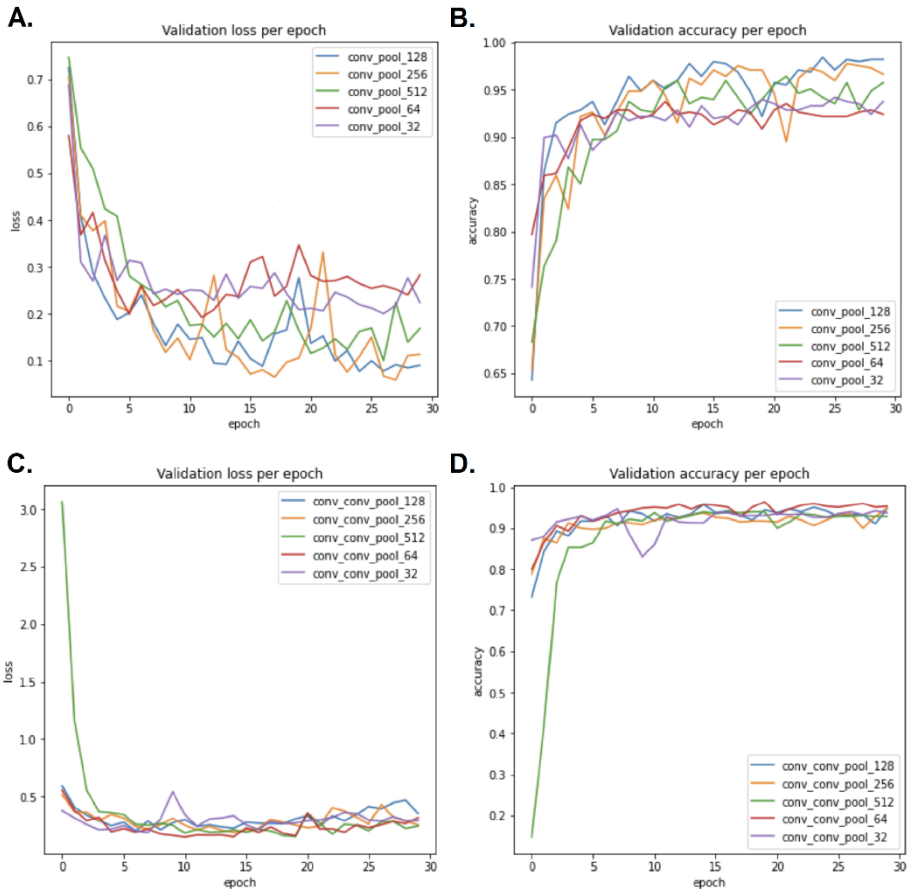


Figure 11 – Courbe de perte de validation des différents modèles répétant une couche de convolution avec une couche de pooling en fonction de l'epoch (A.) et courbe de précision de validation selon les epochs (B.). Courbe de perte de validation des différents modèles répétant deux couches de convolutions et une couche de pooling (C) et courbe de précision de validation selon les epochs (D)

Le modèle conv_pool_128 atteint la meilleure précision et fut sélectionné pour la répétition d'une couche de convolution et d'une couche de pooling (Figure 11B.) et le modèle conv_conv_pool_64 pour la répétition convolution-convolution-pooling (Figure 11D.).

L'influence de la couche cachée pour ces deux modèles a été ensuite analysée de la même manière en répétant le nombre de couches dense ainsi que leurs nombres de neurones.

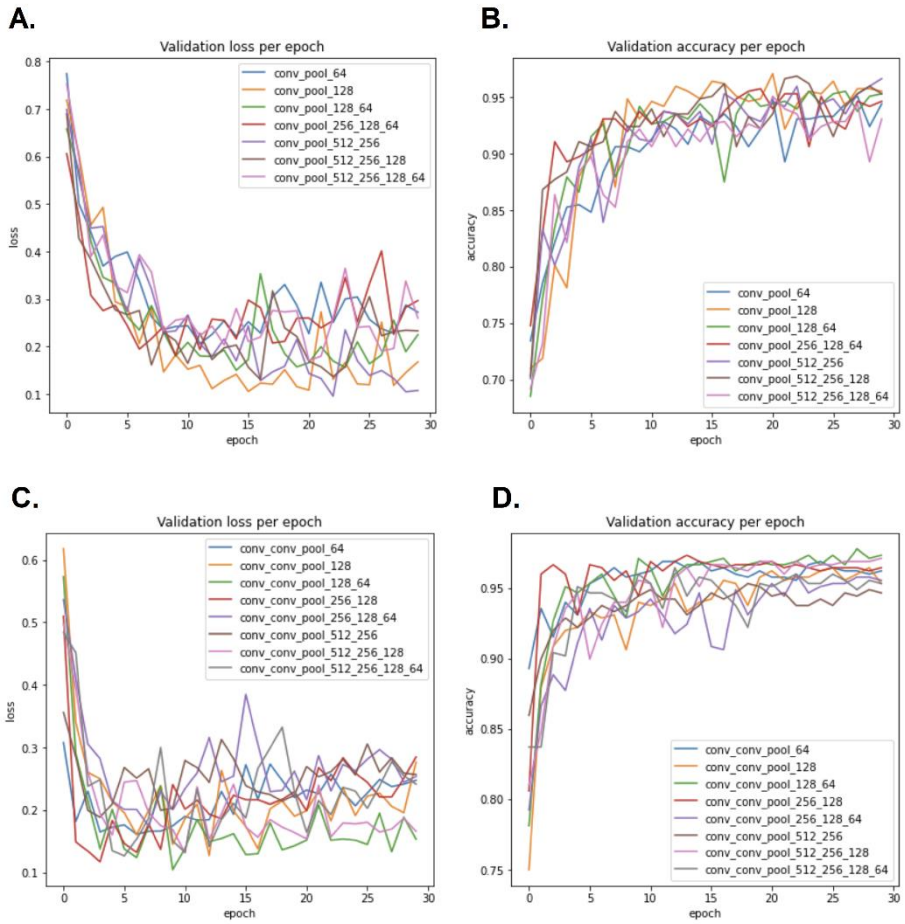


Figure 12 – Courbe de perte de validation du modèle conv-pool sélectionné avec différentes architectures de couche cachée (A.) et leurs courbes de précision de validation selon les epochs (B.). Les couches denses analysées sont décrites par les nombres en suffixe. Exemple : conv_pool_128_64 représente le modèle conv-pool sélectionné avec une couche cachée constituée d'une couche dense de 128 neurones suivie d'une couche dense de 64 neurones. Courbe de perte de validation des différents du modèle conv-conv-pool sélectionné avec différentes architectures de couche cachée (C.) et leurs courbes de précision de validation selon les epochs (D.).

Pour le modèle convolution-pooling, la couche cachée sélectionnée est celle constituée d'une unique couche dense de 128. Le modèle est nommé « conv_pool_128 » sur la figure 12A et 12B. Il atteint la meilleure précision sur le jeu de validation. Pour le modèle de convolution-convolution-pooling, la couche cachée sélectionnée est celle constituée d'une première couche dense de 128 neurones suivie d'une seconde couche de 64 neurones. Le modèle est nommé « conv_conv_pool_128_64 » sur les figures 12C et 12D.

Afin d'éviter le surapprentissage, une couche de dropout après chaque couche de pooling a été ajoutée, ainsi qu'une couche de BatchNormalization après chaque couche convolutive. Les meilleurs hyperparamètres de ces deux architectures ont été ensuite cherchés à l'aide la librairie Keras Tuner [O'Malley2019]. Les hyperparamètres qui ont été affinés sont les suivants :

- Couche convolutive : Nombre de filtres et taille des kernels
- Dropout : Taux

- Couche dense : Nombre d'unités
- Optimisateur Adam: Taux d'apprentissage

En utilisant la classe Hyperband de Keras Tuner, une recherche aléatoire des combinaisons est effectuée en débutant par scanner le champ des possibilités sur un très petit nombre d'epochs, pour ensuite continuer avec les modèles les plus prometteurs. Pour l'architecture convolution-pooling, le meilleur modèle obtenu obtient un score de précision d'approximativement 0.962 sur le jeu de test, et celui obtenu pour l'architecture convolution-convolution-pooling obtient approximativement 0.997. Le modèle convolution-convolution-pooling avec les hyperparamètres affinés est ensuite entraîné sur tout le jeu de données d'entraînement. Le modèle final obtenu obtient une précision de 0.986 sur l'ensemble du jeu de test (Figure 13).

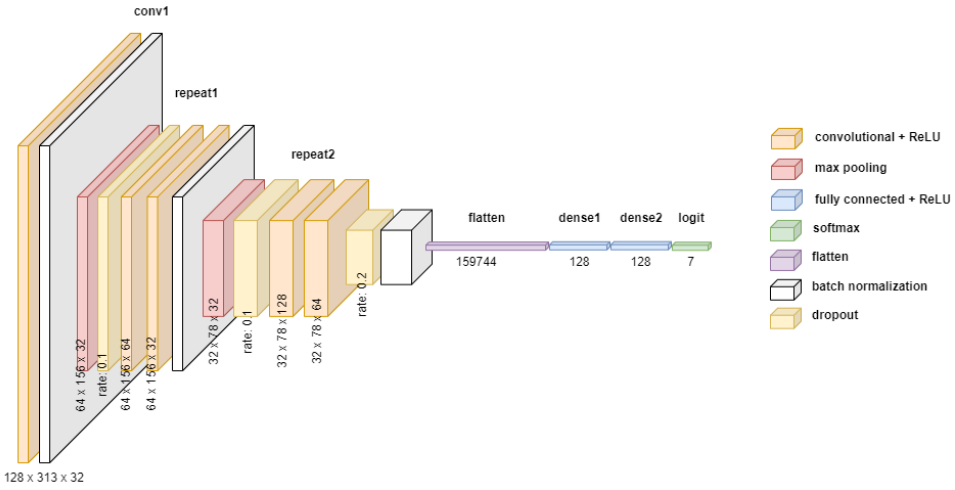


Figure 13 – Architecture du réseau neuronal convolutif final obtenu pour la tâche auxiliaire de classification des types de machines.

d-2) Détection d'anomalies avec les autoencodeurs et les embeddings du classifieur de types de machine:

Notebook : Iteration 2 Experimenting with autoencoders.ipynb

Le développement du classifieur de types de machine avait pour but d'extraire des données spécifiques aux échantillons audios des différentes machines. Les données compressées des couches intermédiaires de ce réseau neural convolutif (CNN) ont été ensuite utilisée comme donnée d'entrées pour des auto-encodeurs. Dans cette itération le but a été de créer un auto-encodeur pour chaque type de machine et de les entraîner sur le jeu de donnée d'entraînement contenant les échantillons normaux afin de reconstruire le plus fidèlement les données d'entrées. Cette modélisation se base sur l'hypothèse que les échantillons anormaux sont en-dehors de la distribution des échantillons normaux et que lors de la reconstruction par l'auto-encodeur spécifique, la perte serait plus élevée pour les anomalies.

Trois couches intermédiaires du classifieur de types de machine a été utilisé comme données d'entrée pour les auto-encodeurs :

- La dernière couche de convolution
- La couche intermédiaire de pooling

- La première couche dense de la couche cachée

Les auto-encodeurs utilisaient la fonction de perte « mean squared error » mse et ils étaient composée (Figure 14) :

- Une couche d'entrée
- Une couche dense de 128 neurones
- Une couche dense de 64 neurones
- Une couche dense de 32 neurones
- Une couche dense de 32 neurones
- Une couche dense de 64 neurones
- Une couche dense de 128 neurones
- Une de sortie au même format que la couche d'entrée

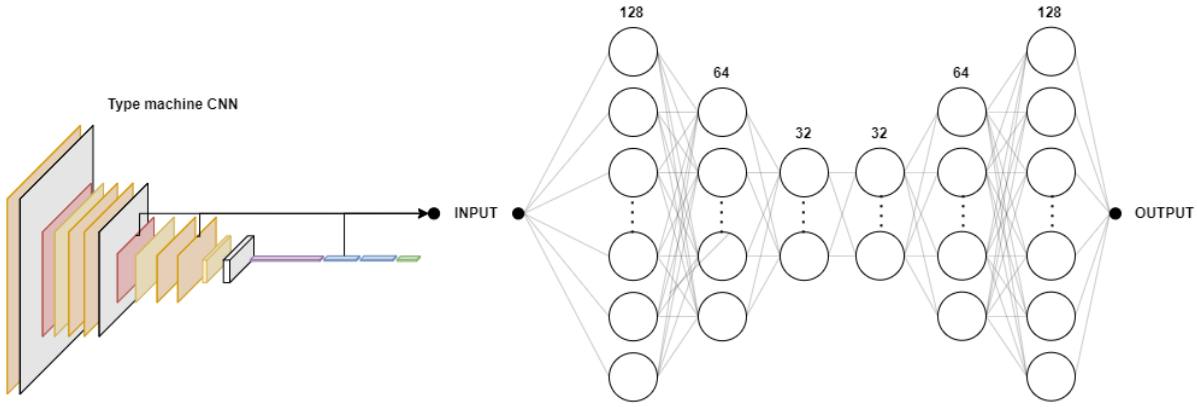


Figure 14 – Architecture des auto-encodeurs prenant en entrée les embeddings de différentes couches intermédiaires du classifieur de type de machine.

Après l'entraînement des auto-encodeurs, les pertes des échantillons normaux ont été calculées et le 95ème percentile a été utilisé comme seuil. Les pertes plus élevées que ce seuil sont considérés des anomalies. Les scores AUC de la courbe ROC ont été ensuite mesurées.

Pour la dernière couche de convolution et la dernière couche intermédiaire les scores AUC stagnaient à 50%. Pour la première couche dense, les ventilateurs (fan) et les rampes (slider) présentaient des scores plus élevés que l'aléatoire :

	Valve	Bearing	Fan	Gearbox	Slider	ToyCar	ToyTrain
AUC	0.48%	0.49%	0.64%	0.51%	0.61%	0.50%	0.48%

Tableau 3: Score AUC des différents auto-encodeurs spécifiques à un type de machine en utilisant les embeddings de la première couche dense du classifieur de machine.

En appliquant une analyse principale des composants sur les embeddings de la première couche dense on constate que les sons normaux et anormaux ne sont pas écartés et qu'ils sont localisés dans la même distribution (Figure 15).

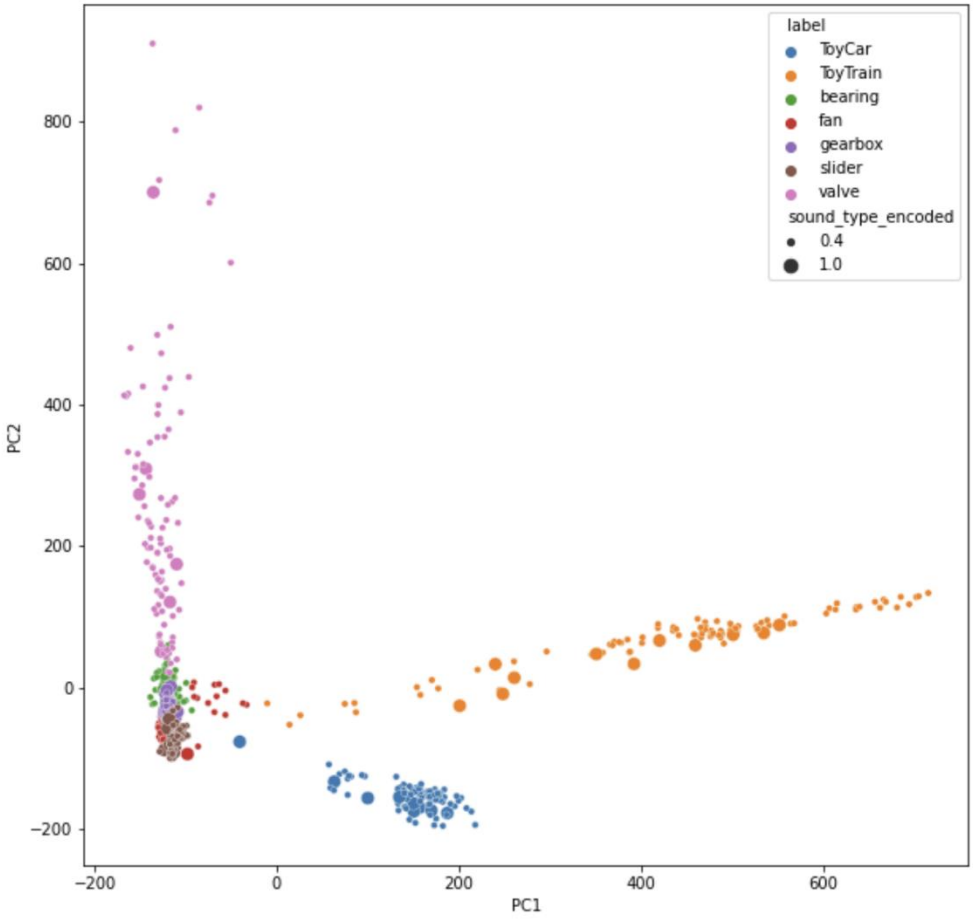


Figure 15 – Analyse des composantes principales des embeddings de la première couche dense du classifieur de type de machine. Le label « sound_type_encoded » correspond aux sons normaux de petite taille (0.4) et aux sons anormaux de plus grande taille (1.0)

d-3) Détection d'anomalies avec les embeddings d'un classifieur de section :

Notebook : Iteration 3 Select CNN section ID clf with center loss.ipynb

À partir de cette itération les résultats du challenge DCASE 2022 étaient sortis. L'équipe de Kuroyanagi & al. a obtenu la seconde place et ont proposés des modèles spécifiques aux types de machine combinant deux fonctions de pertes de types entropie binaire croisée [Kuroyanagi2022]. La première perte permettait d'identifier à quelle section appartenait l'échantillon audio et la seconde permettait de classifier l'échantillon comme un son provenant du type de machine ou d'une autre (classé pseudo-anormal). L'idée principale de ce modèle était de séparer les sons qui se ressemblaient provenant de la même machine, ainsi que de séparer les sons provenant des autres machines.

C'est dans la même optique que durant cette itération, des modèles spécifiques aux machines ont été modélisés afin de les classifier selon leurs sections. Le but étant d'obtenir un écart entre les échantillons audio provenant de différentes sections, une fonction de perte centerloss fut appliquée simultanément avec une perte de type entropie croisée catégorique. L'hypothèse durant cette itération était qu'en accentuant la séparation des sections, l'écart entre sons normaux et sons anormaux serait aussi plus accentué.

$$Categorical\ cross\ entropy\ =\ -\sum_{i=1}^{m}\sum_{j=1}^{nb\ classes} y_{ij}\,.\log\left(y_{ij}\right)$$

$$Centerloss\ =\ \frac{1}{2}\sum_{i=1}^{m}\quad \|\ x_i\ -\ c_{yi}\ \|^2$$

$$Total\ loss\ =\ \sum_{i=1}^{m}\left(\ -\sum_{j=1}^{nb\ classes}\left(y_{ij}\cdot\log\left(\dot{y_{ij}}\right)\right)\ +\ \lambda\ \cdot\|\ x_i\ -\ c_{yi}\ \|^2\right)$$

$$=\sum_{i=1}^{m}\left(CategoricalCrossEntropy\ +\ \lambda\cdot CenterLoss\right)$$

Avec m représentant le nombre d'échantillons, y la classe réelle de l'échantillon, $\dot{y}$ la prédiction fait par le modèle de l'échantillon, c le centroïde de la classe de l'échantillon. À chaque itération de l'entraînement, la fonction perte centerloss va redéfinir les coordonnées des centroïdes en calculant la différence entre les échantillons et leurs centroïdes spécifiques. Cette différence est ensuite multiplié par un facteur α et le produit est retiré des anciennes coordonnées des centroïdes. La convergence est atteinte lorsque les centroïdes obtiennent des coordonnées minimisant les distances entre eux et les échantillons appartenant à la même classe. Le lambda λ permet de contrôler l'importance de la fonction de perte centerloss. Ce sont les embeddings de la couche dense du CNN qui sont utilisés pour cette fonction perte.

L'architecture du classifieur est décrite dans la figure 16. Les données d'entrées sont les embeddings obtenus à partir d'une couche intermédiaire du classifieur de type de machine modélisé auparavant. Pour chaque type de machine un ensemble de modèles ont été entraînés avec des lambdas pour la fonction centerloss allant de 1e-2 à 1e-11 sur un ensemble de 1200 sons du jeu d'entraînement sélectionnés pour obtenir une distribution uniforme entre les sections.

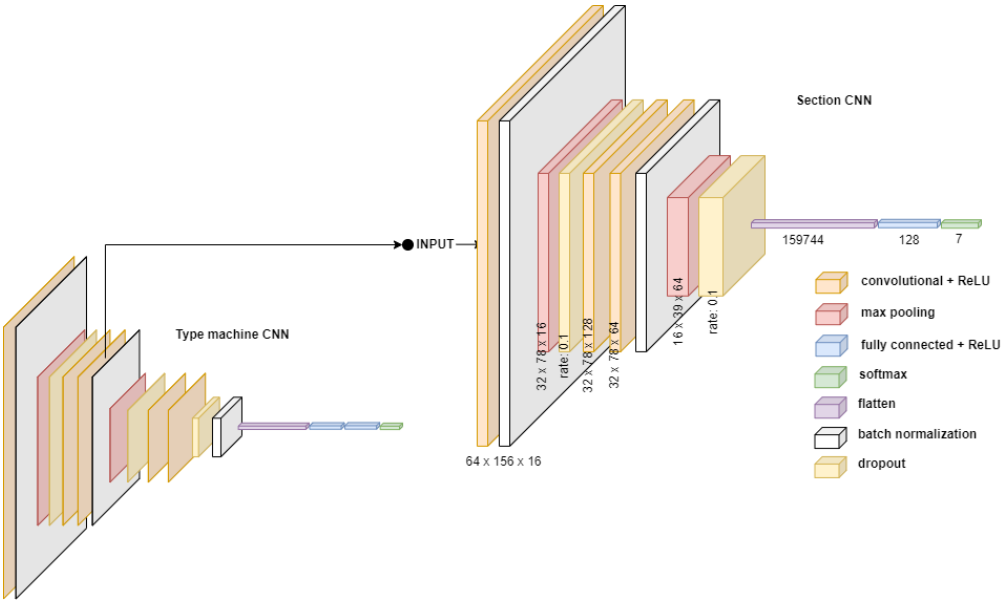


Figure 16 – Architecture des réseaux neuronaux convolutifs pour la classification des sections d'un type de machine.

Pour visualiser les données compressées de la dernière couche dense des modèles, des techniques de réduction PCA, t-SNE et ISOMAP ont été effectuées (Figure 17). Les embeddings obtenus pour le ventilateur présente une bonne séparation entre les sections pour les échantillons normaux mais semble aussi séparer les anomalies des distributions normales. Cette séparation n'était visible que pour certaines valeurs de lambda et était moins nette pour certaines machines. (Les figures peuvent être retrouvées dans le notebook de la section).

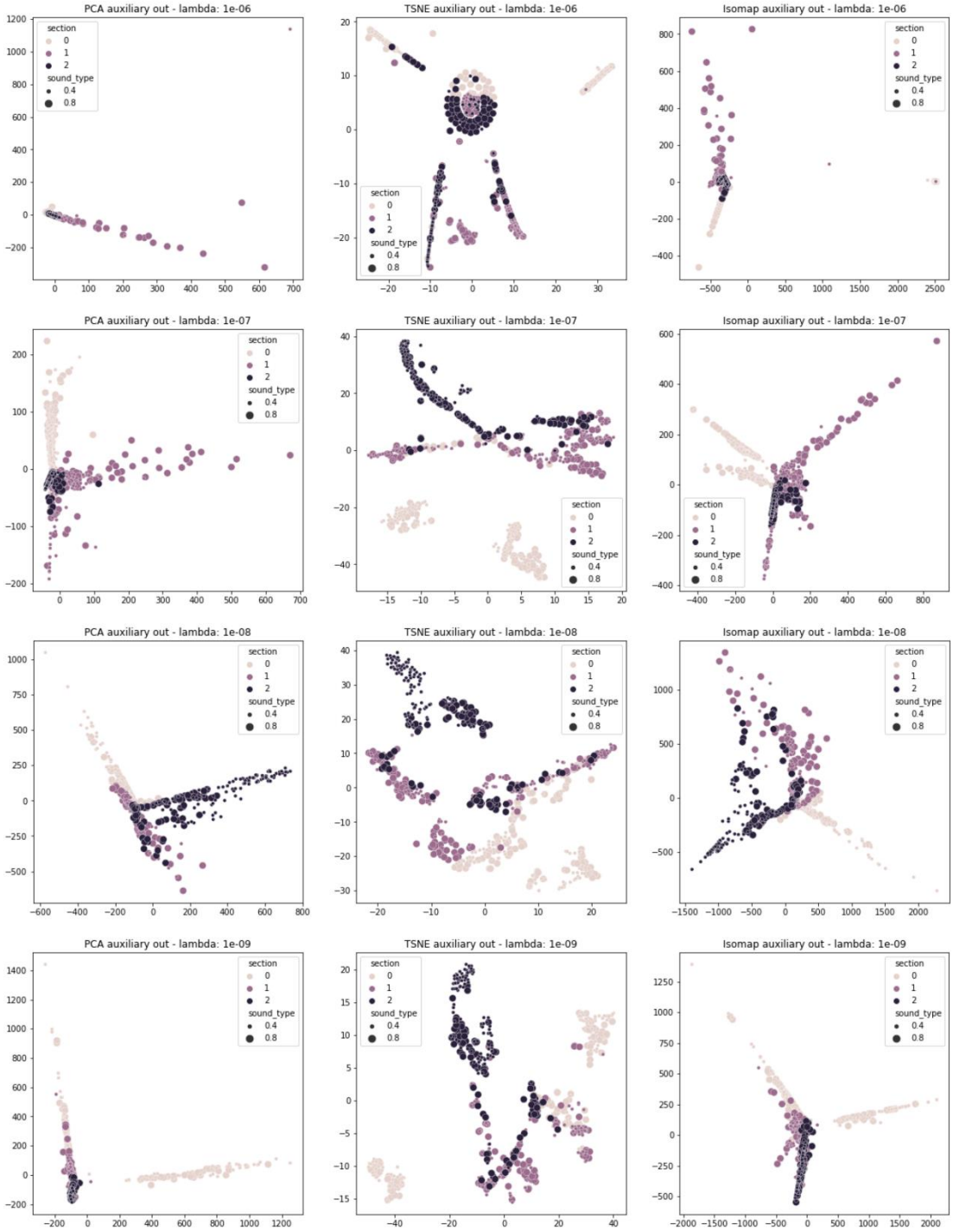


Figure 17 – Visualisation des techniques de réduction sur les embeddings de la couche dense de 128 unités du CNN classifiant les sections d'échantillons provenant de ventilateur. De gauche à droite : PCA, t-SNE, ISOMAP et de bas en haut des lambda qui régulent le poids de la fonction de perte centerloss variant de 1e-06 à 1^{e}-09. Le « sound_type » correspond à la normalité des échantillons avec la taille 0.4 correspondant aux sons normaux et 0.8 aux sons anormaux.

Les modèles avec les lambdas les plus prometteurs selon une inspection visuelle ont été sélectionnées afin d'utiliser les embeddings produites par la couche dense comme données d'entrées pour les auto-encodeurs. Les auto-encodeurs sont composés :

- Une couche dense de 128 unités
- Une couche dense de 64 unités
- Une couche dense de 64 unités

- Une couche dense de 128 unités

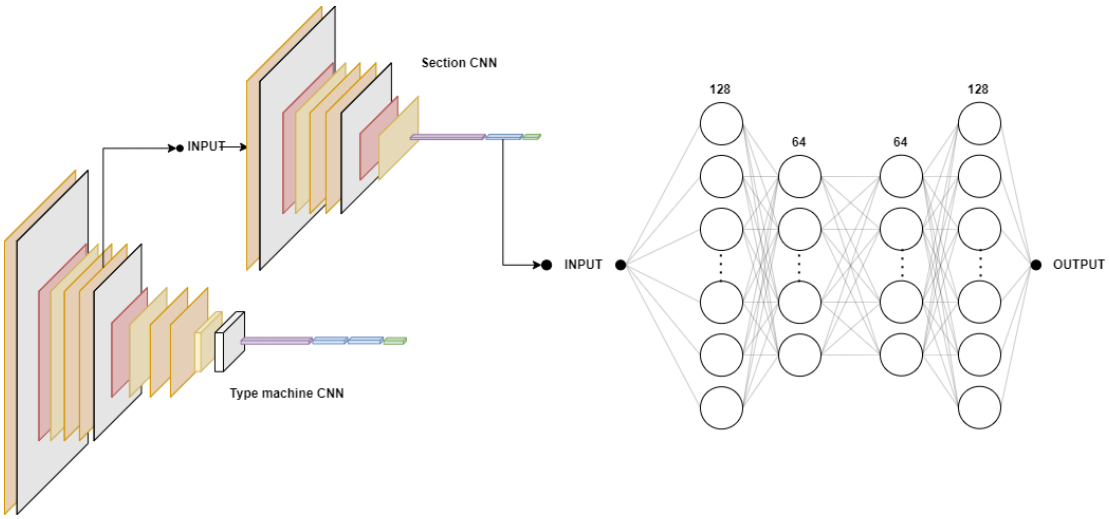


Figure 18 – Architecture des auto-encodeurs prenant en entrée les embeddings de l'unique couche dense de 128 neurones des CNNs classifiant les sections.

Les auto-encodeurs furent entraînés sur une sous-partie des échantillons normaux du jeux de données afin d'accélérer la procédure et d'obtenir un premier retour rapidement. La perte mse fut utilisée en comparant les données entrées et les données reconstruites. Un seuil a été obtenu en calculant une liste de pertes provenant d'échantillons normaux et le 95ème percentile fut sélectionné. Les échantillons obtenant une perte plus élevée que ce seuil sont considérés des anomalies. Les scores AUC des ROC furent calculées. Les valeurs obtenues varient entre 47% et 54% et cette méthode est donc encore moins performante que celle utilisant uniquement les classifieur de type de machine avec les auto-encodeurs. Une alternative aux auto-encodeurs a été d'utiliser la fonction de perte centerloss et de mesurer la distance des échantillons selon le centroïdes de leurs sections. Cette méthode ne fut pas plus performante. Une modélisation en mixture de distribution Gaussienne des embeddings du classifieur de sections a aussi été effectué, sans accroissement de performance.

En employant des techniques de réductions afin de visualiser en deux dimensions les données compressées utilisées par les auto-encodeur, les erreurs de l'ensemble des modèles étaient que les échantillons anormaux se situaient dans la distribution des échantillons normaux. Malgré les tentatives d'utilisés une deuxième tâche auxiliaire de classification avec une fonction de perte centerloss, il n'y a pas eu d'améliorations dans la performance du système. Le problème survient aussi certainement à la mauvaise idée d'utiliser les embeddings du classifieur de type de machine pour mettre en séquence un deuxième classifieur pour les sections, car les embeddings de ce premier n'ont permis que d'obtenir des scores AUC de 60% lors de la détection d'anomalies par des auto-encodeurs.

Classification supervisée (labels normal/anormal connus pour l'entraînement)					
Data	**Input**	**Pre-processing**	**Modèle**	**Résultats**	**Notebook**
500 sons normaux + 300 anormaux par machine (section, domaine, attributs aléatoires)	Spectrogrammes (n_fft = 1024, hop_length = 512, échelle linéaire) aplatis en vecteurs	SelectPercentile (30%) + PCA(85%)	XgBoost (avec tuning des hyperparamètres)	Voir Fig. 6	ASD_supervised_clf_sounds_GF.ipynb
idem	Features issues des coefficients de l'ondelette	aucun	GradientBoostingClassifier	0.65 <AUC <0.90	machine_learning_wavelet_gradientboost_SD
600 sons normaux + 300 anormaux par machines tous regroupés	STFT	aucun	CNN avec softmax / accuracy en metrics	Voir Fig. 7	modele DL supervised with fourrier transform_SD
450 sons normaux (stratifié par section) + 300 anormaux, uniquement pour le type de machine valve	Mel-spectrogrammes 32 filtres mel * 32 frames (n_fft=1024, hop_length=512, n_mels=32)	aucun	ANN avec sigmoïde	Acc: 0.6, càd non significatif	Iteration 0 Simple model
Classification non supervisée (labels normal/anormal inconnus pour l'entraînement)					
Data	**Input**	**Pre-processing**	**Modèle**	**Résultats**	**Notebook**
Tout le train set pour l'entraînement (et tout le test set pour la prédiction du label normal/anormal)	Mel-spectrogrammes 2D 128x313 (n_fft = 1024, hop_length = 512, n_mels = 128)		Modèles DL basiques type CNN+DNN entrainés à reconnaitre le type de machine	Voir Fig. 8	ASD_clf_sounds_DL_from_machine_clf_GF.ipynb
Train set et test set par section (0, 1, 2), uniquement dans le domaine source	Idem		Idem	Voir Fig. 9	ASD_clf_sounds_DL_from_machine_clf_GF.ipynb
Train set et test set par machine	Idem		Modèles DL basiques type CNN+DNN entrainés à reconnaitre la section	AUC=0.71 pour la gearbox (autres AUCs inférieures)	ASD_clf_sounds_DL_from_section_clf_GF.ipynb
Train set et test set par machine	Idem		FaceNet + triplet loss sur la section → vecteurs d'embedding de taille 128 Random Forest	Voir Fig. 10	ASD_clf_sounds_FaceNet_section_GF.ipynb
1000 sons du train set par machine et tout le test set	Idem		FaceNet + triplet loss sur la machine → vecteurs d'embedding de taille 128 Random Forest	Voir Fig. 10	ASD_clf_sounds_FaceNet_machine_GF.ipynb
Train set et test set par machine	STFT/ log(STFT)		Auto-encodeurs à architecture classique ou avec multi-décodeurs	Non significatif	Tries_with_autoencoders_SD & Multi-decodeur based suffixe DL for

					anomaly detection (other tries part 2)
Train set et test set par machine + slices des audios en 5 morceaux de 2 secondes	STFT		Modèles DL qui décompose le son original en 5 parties	Non significatif	Split sound for DL unsupervised & split and suffix for DL
Tout le train set pour l'entraînement et tout le test set pour l'évaluation	Mel-spectrogrammes 2D 128x313 (n_fft = 1024, hop_length = 512, n_mels = 128)		Autoencodeurs utilisant les embeddings de couches intermédiaires d'un CNN classifiant les types de machines	AUC=0.64 pour fan. AUC=0.61 pour slider	Iteration 2 Experimenting with autoencoders
1200 sons du train set pour l'entraînement et tout le test set pour l'évaluation	Mel-spectrogrammes 2D 128x313 (n_fft = 1024, hop_length = 512, n_mels = 128)		Autoencodeurs utilisant les embeddings de couches intermédiaires de CNNs classifiant les sections avec ou sans fonction de perte centerloss	Non significatif	Iteration 3 Select CNN section ID clf with center loss

Tableau 4– Principales méthodes, discutées dans ce rapport, de classification des sons entre sons normaux et anormaux. D'autres approches ont été testées mais elles ne sont pas listées ici car elles ont été soit infructueuses, soit désignées à des tâches annexes.

Description des travaux réalisés

Répartition de l'effort sur la durée et dans l'équipe

Voir le diagramme de gantt (en annexe) et le tableau récapitulatif des principales méthodes en page précédente.

Bibliographie

[Velardo2020] Cours *"Audio signal processing for machine learning"* (2020-2021) Chaine YouTube de Valerio Velardo (lien)

[Rothmann2018] *"What's wrong with CNNs and spectrograms for audio processing?"* Post de Daniel Rothmann sur le site « Towards Data Science » (lien)

[Müller2021] *"Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning"* Robert Müller, Fabian Ritz, Steffen Illium, Claudia Linnhoff-Popien Proceedings of the 13th International Conference on Agents and Artificial Intelligence, Vol. 2, p. 49 (2021) – arXiv:2006.03429

[Diallo2019] Diallo, M. S., Mokeddem, S. A., Braud, A. & Frey, G. « *Quels jeux de données pour la prédiction d'anomalies dans l'industrie 4.0 ?* ». (lien)

[Vaulpane2018] « *Pourquoi la maintenance prédictive va-t-elle révolutionner l'industrie ?* » J.R. de Vaulpane. Article dans les Echos. (lien)

[Dohi2022] « Description and Discussion on DCASE 2022 Challenge Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalization Techniques. » Dohi, K. et al. (lien)

[Chen2022] « Learning to Adapt to Domain Shifts with Few-shot Samples in Anomalous Sound Detection. » 2022 Chen, B., Bondi, L. & Das, S. (lien).

[O'Malley2019] « Keras Tuner » 2019 O'Malley, Tom and Bursztein, Elie and Long, James and Chollet, François and Jin, Haifeng and Invernizzi, Luca and others (lien)

[Kuroyanagi2022] « Two-stage anomalous sound detection systems using domain generalization and specialization techniques. » 2022 I. Kuroyanagi, T. Hayashi, K. Takeda and T. Toda (lien)

[PyWav] Guide pour l'utilisation des wavelets avec la librairie PyWavelets (lien)

Bilan & Suite du projet

Bilan par rapport aux objectifs

Les objectifs fixés n'ont pas été atteints mais plusieurs conclusions intéressantes ont été obtenues :

- la classification *supervisée* des sons normaux/anormaux donne de bons résultats (0.8<AUC<0.98 selon machines) avec des approches très simples de machine learning. De plus, nous n'avons passé qu'une semaine (en parallèle des cours, donc typiquement 2 jours) sur cette tâche et il est donc fort probable que l'on aurait pu améliorer ces résultats.
- la tâche de classification *supervisée* des sons normaux/anormaux n'a pas été améliorée en considérant des modèles de deep learning. Toutefois, cette étape a été réalisée à un moment où nous faisions nos tous premiers pas en deep learning et nous n'avons pas tenté de revenir *a posteriori* sur ce problème puisque les approches machine learning s'étaient avérées efficaces. De plus, le réel objectif du challenge concernait la classification non supervisée.
- la classification *supervisée* des types de machine s'est avérée être assez « facile » en deep learning, dans le sens où un modèle simple sans optimisation fournit déjà une précision moyenne de 97% sur le jeu de test. En cherchant à optimiser ce modèle, nous sommes parvenus à obtenir une précision moyenne supérieure à 98% avec une architecture convolution-convolution-pooling.
- la tâche de classification *non supervisée* des sons normaux/anormaux s'est avérée difficile. Les résultats obtenus pour l'AUC sont fortement dépendant du type de machine et de la section. En considérant toutes les sections du jeu de test, nous avons obtenu notre meilleur AUC pour le slider ($AUC \approx 78\%$) avec une architecture assez simple à base de couches de convolution, batch normalization et dense, en utilisant les mel-spectrogrammes 2D bruts de taille 128x313 en entrée. Ce résultat est bien en deçà de l'AUC obtenue par l'équipe gagnante du challenge ($AUC \approx 90\%$). En considérant uniquement le domaine source (conditions plus faciles du challenge DCASE2020), section par section, nous avons obtenu notre meilleur AUC pour le ToyCar dans la section 2 ($AUC \approx 90\%$), à comparer aux AUCS données comme benchmarks sur la page web du challenge 2022 : $AUC \approx 99\%$ avec une méthode par autoencodeurs et $AUC \approx 74\%$ par une approche MobileNetV2 (voir tableau en bas de page, nous n'avons pas poussé la comparaison aux autres valeurs). Nous avons également tenté de multiples approches par auto-encodeurs mais aucune d'entre elles ne s'est avérée efficace pour la tâche de classification *non supervisée.*

Contribution à l'accroissement des connaissances scientifiques

- Si les échecs sont considérés comme un accroissement scientifique alors l'ensemble couplant séquentiellement un CNN classifiant les types de machine avec des CNNs classifiant les sections pour obtenir des embeddings utilisés par des auto-encodeurs est une méthode qui ne fonctionne pas de la manière où elle a été appliquée ici (Figure 17).

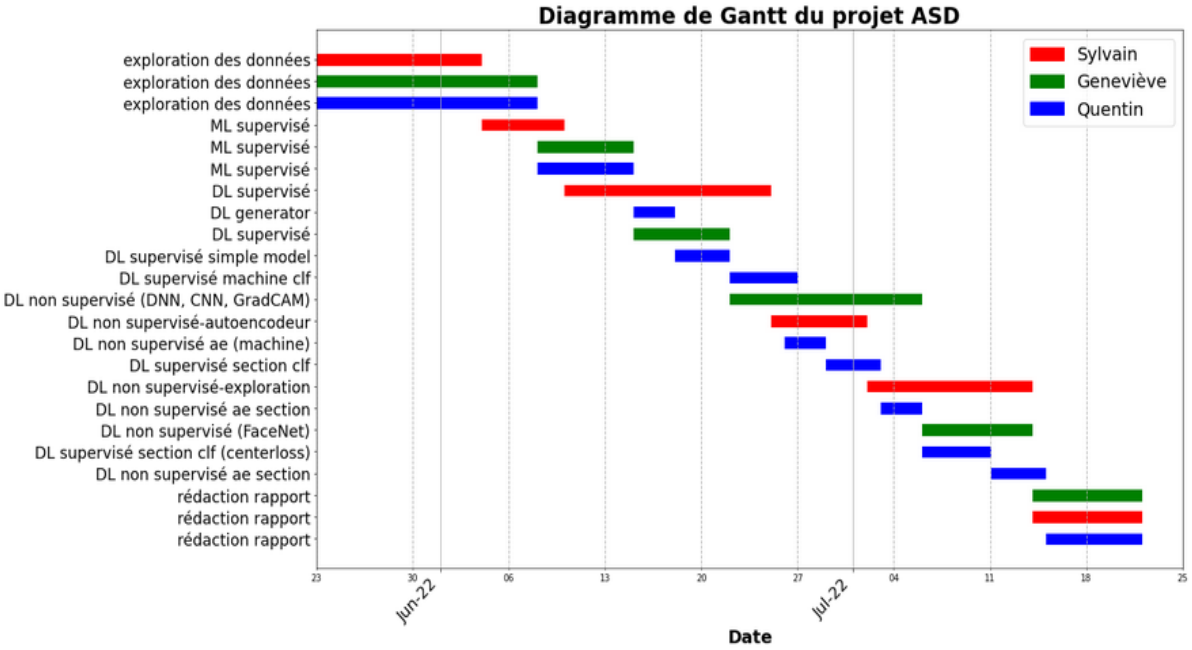
Suggestions de pistes d'amélioration

- **Faire du « transfer learning » :** des modèles adaptés aux problèmes audio sont par exemple disponibles sur la plateforme TensorFlow Hub (lien) ou sur la page du challenge DCASE 2022 (lien). Cette stratégie a été notamment utilisée dans l'article [Müller2021].
- **Ecouter les sons et analyser à l'œil les spectrogrammes du jeu de test connaissant nos prédictions « son normal/son anormal » :** ceci aurait permis de visualiser si les sons mal prédits correspondent à des spectrogrammes dont l'anomalie n'est pas visible à l'œil et/ou à l'oreille.
- Faire une classification binaire « normal/anormal » par machine **en utilisant comme sons anormaux les sons des autres machines**.
- Au lieu d'entraîner à reconnaître le type de machine ou la section**, entraîner à reconnaître le couple (machine, section)**, soit 21 classes au total ; **ou les attributs** (mais ils sont sous-représentés dans le domaine cible du jeu d'entraînement).
- **Dans l'approche FaceNet** fournissant des « embedding vectors » de taille 128, tester différentes fonctions coûts (en jouant notamment sur la marge de la triplet loss) et compléter par une approche de clustering (KMeans, Mean Shift, …) dans l'idée que si un son est trop éloigné de son centroïde, il est sans doute anormal. Ceci serait pertinent dans les cas où l'analyse en PCA des « embedding vectors » montre des distributions disjointes des sons normaux et anormaux.
- **Faire une analyse GradCAM** (commencée pendant le projet mais abandonnée) pour identifier les zones pertinentes des spectrogrammes et éventuellement ainsi tronquer certaines zones afin de réduire la dimensionalité.
- **Utiliser une approche d'ensemble :** nos résultats semblent montrer qu'il n'existe pas une unique méthode qui donne les meilleurs résultats pour tous les types de machines ou tous les types de section. Ainsi, il paraît judicieux de choisir des méthodes optimisées différentes par machine/ou et section.
- **Approfondir le pré-processing des spectrogrammes :** par exemple, appliquer des filtres haute/basse-fréquence ou des filtres de convolution à noyau fixe pour faire ressortir certaine caractéristiques. Le filtre devra sans doute dépendre du type de machine (les spectrogrammes étant à l'œil qualitativement très différents entre machines). Il serait aussi intéressant d'appliquer des filtres gammatone reproduisant le fonctionnement de détection d'un son par l'humain de l'oreille au cerveau.
- **Utilisation d'autres formats de données d'entrées pour les modèles.** Ici nous avons principalement utilisé les mel-spectrogrammes avec 128 filtres mel et 313 frames, une frame correspondant à 64ms. L'ensemble du mel-spectrogramme était procuré aux modèles. Les organisateurs et la majorité des participants ont utilisés en tant que données d'entrées des frames consécutives avec un saut de frames entre chaque séquence consécutives. Ces données permettraient d'analyser une intervalle temporelle plus grande et d'analyser des frames plusieurs fois. Elles demandent aussi un coût computationnel qu'il aurait été difficile d'assumer avec nos ordinateurs.
- **Estimer le bilan carbone de nos codes** (par exemple avec l'outil https://codecarbon.io/). Ici, nous avons abordé le projet comme un challenge avec l'objectif d'obtenir les meilleurs AUCs avec nos puissances de calcul. Dans le contexte d'urgence climatique actuel, il est aussi important de réfléchir à une utilisation raisonnée de l'IA compte-tenu de l'objectif à atteindre pour tel ou tel projet, sans forcément chercher l'optimisation absolue si cela s'accompagne d'une augmentation forte de l'empreinte carbone. De façon générale, un

client industriel sera également intéressé à limiter ses factures énergétiques liées aux machines de calcul et/ou à développer des solutions d'IA embarquées.

- **Utilisation de la fonction perte centerloss sur le classifieur de type de machine.** Un des ensembles de modèles mettaient séquentiellement un classifieur de type de machine, un classifieur de section et un autoencodeur. La fonction centerloss a uniquement été utilisée sur les classifieurs de sections qui utilisaient des embeddings du classifieur de type de machine, ces données pouvant peut-être ne plus contenir les informations nécessaires. En utilisant la fonction centerloss et en forçant le CNN à séparer fortement les types de machine, l'hypothèse est que les features extraites seraient plus spécifiques aux machines.

Annexes



Description des fichiers de code

Notebooks de Sylvain:

- **sound_features_exploration_SD :** Contient les codes nécessaires pour visualiser les différentes propriétés d'un enregistrement audio avec librosa, de la transformée de Fourrier au centroïde spectral en passant par le spectre amplitude/fréquence et le Mel.
- **machine_learning_wavelet_gradientboost_SD :**Contient un premier modèle de machine learning supervisé utilisant les caractéristiques des ondelettes de Daubechies pour déterminer si le son est normal ou pas. Cela fonctionne par machines en utilisant le classifieur « Gradient Boosting ».
- **modele DL supervised with fourrier transform_SD:** Contient les essais deep learning supervisé en utilisant des réseaux de convolution 1D ou 2D, celui en 2D est poussé pour analyser les sons et les classifier par type de machines s'ils sont normaux ou apparaissent dans une catégorie commune (la classe 0) s'ils sont considérés comme anormaux.
- **Tries_with_autoencoders_SD :** Contient des essais d'auto-encodeurs basés sur l'encodage/décodage des transformées de Fourrier (logarithmique ou non). Malheureusement, n'aboutissent pas à des résultats corrects.
- **Other tries with unsupervised Deep learning (part 1):** Contient d'autres essais avec les transformées de fourrier mais ne donne pas de résultats concluants
- **Multi-decodeur based suffixe DL for anomaly detection (other tries part 2):** Contient un essai ou l'encodage vers les suffixes des machines et commun à tous les suffixes puis on entraine un décodeur par type de suffixes, enfin les sons passent par tous les décodeurs pour déterminer si le son est normal en additionnant les mse trouvé par décodeurs. Ne fonctionne pas comme espérer. La suite reprend plusieurs mse de plusieurs méthodes pour essayer de les assembler (recherche de meilleur discriminant que l'erreur quadratique moyenne (mse) classique) afin d'établir un seuil correct pour la détection d'anomalies. Pas de résultats utilisables.
- **Split sound for DL unsupervised & split and suffix for DL :**Contient une méthode pour générer un dataframe contenant les sons découpés en 5 parties de 2 secondes reliés chacune à l'audio original. Utilisé seul ou en complémentaire des suffixes (qui caractérisent un élément de l'audio) comme « bottleneck » d'un network qui encode le son original en ses 5 parties. N'a pas fonctionné pour reconnaitre les anomalies. Revoir le réseau de neurones ainsi que faire varier les features extraites des audios pourraient conduire à de meilleurs résultats.

Notebooks de Geneviève :

- **ASD_dataviz_GF.ipynb :** notebook permettant d'analyser l'organisation du jeu de données, écouter les sons, visualiser différents spectrogrammes et les comparer 2 à 2. Il a notamment été utilisé pour réaliser les figures 2, 3,4 et 5.
- **ASD_supervised_clf_sounds_GF.ipynb :** notebook de classification supervisée des sons (normaux/anormaux) par machine learning. Différentes méthodes de réductions et différentes classifieurs ont été testés. Un tableau bilan est fourni à la fin. Ce notebook a notamment été utilisé pour réaliser la figure 6.
- **ASD_supervised_clf_machines_GF.ipynb :** notebook de classification supervisée des machines par machine learning (PCA + xgboost). Ecrit rapidement pour vérifier que l'approche Deep Learning était plus pertinente pour cette tâche (mais la comparaison n'a pas été poussée jusqu'au bout).

- **ASD_supervised_clf_sounds_DL_GF.ipynb :** notebook de classification supervisée des sons (normaux/anormaux) par deep learning. Plusieurs types de spectrogrammes ont été testés en entrée et des réseaux de neurones de type LeNet ont été considérés. Les résultats sont très mauvais (tous les sons prédits comme normaux ou tous prédits comme anormaux), sauf dans un cas où l'on obtient un AUC=0.59 pour la gearbox. Les autres types de machine n'ont pas été considérés. Ce notebook mériterait à être repris car il a été écrit avec très peu de recul sur les modèles de Deep Learning.
- **ASD_clf_sounds_DL_from_machine_clf_GF.ipynb :** notebook de classification non supervisée des sons (normaux/anormaux) par deep learning, en utilisant une tâche annexe de classification des machines. Les données d'entrée sont les mel-spectrogrammes 128x313. Ce notebook a notamment été utilisé pour réaliser les figures 8 et 9.
- **ASD_clf_sounds_DL_GradCAM_GF.ipynb :** notebook complémentaire (version préliminaire) de ASD_clf_sounds_DL_from_machine_clf_GF.ipynb. *Attention, les calculs d'AUC sont faux !* Une analyse GradCAM est effectuée à la fin mais sans succès car le modèle utilisé ici était beaucoup trop simple (une seule couche de convolution). Cela mériterait à être repris.
- **ASD_clf_sounds_DL_from_section_clf_GF.ipynb :** notebook de classification non supervisée des sons (normaux/anormaux) par deep learning, en utilisant une tâche annexe de classification des sections, machine par machine. Les données d'entrée sont les mel-spectrogrammes 128x313.
- **ASD_clf_sounds_FaceNet_machine_GF.ipynb :** notebook de classification non supervisée des sons (normaux/anormaux) par une approche d'embedding FaceNet sur le type de machine puis un classifieur Random Forest. Ce notebook a notamment été utilisé pour réaliser une partie de la figure 10.
- **ASD_clf_sounds_FaceNet_section_GF.ipynb :** notebook de classification non supervisée des sons (normaux/anormaux) par une approche d'embedding FaceNet sur la section, machine par machine, puis un classifieur Random Forest. Ce notebook a notamment été utilisé pour réaliser la figure 10.

<u>Notebooks de Quentin:</u>

- **Preprocessing data into 32x313 melspectrogram.ipynb :** notebook permettant d'obtenir les melspectrogrammes sous format 32 filtres mel x 313 frames.
- **Exploratory Data Analysis.ipynb :** notebook de découverte des données. Découverte des sons et visualisation du taux de croisement, des spectrogrammes, et des chromagrammes (pitch) des différentes machines.
- **Iteration 0 Simple model:** notebook contenant une classification supervisée des anomalies en utilisant différents modèles de réseaux neuronaux denses sur les melspectrogrammes 32*313 des valves.
- **Iteration 1 Data Generators :** notebook contenant les premières tentatives pour créer un générateur de données fonctionnel.
- **Preprocessing data into 313x128 melspectrogram.ipynb :** notebook permettant d'obtenir les melspectrogrammes sous format 313 filtres mel x 128 frames.
- **Iteration 1 CNN classification task.ipynb :** notebook contenant la modélisation de modèles de réseau neuronal de convolution qui a pour but de classifier les sons selon leurs machines d'origines.

- **Iteration 2 Searching for best CNN for classification task.ipynb :** notebook contenant les étapes de modélisation du classifieur de machines qui sera ensuite utilisé pour les premiers autoencodeurs. Utilisation d'hypermodèles avec keras tuner.
- **Iteration 2 Experimenting with autoencoders.ipynb :** notebook contenant les premières expérimentations avec les autoencodeurs en utilisant les embeddings provenant du CNN obtenu dans le notebook ci-dessus.
- **Iteration 3 Select CNN section ID clf with center loss.ipynb :** notebook contenant une première partie de modélisation de CNN spécifique à chaque type de machine afin d'identifier la section d'origine d'un échantillon. La première partie contient de même la modélisation d'autoencodeurs utilisant les embeddings des CNNs et une visualisation des embeddings par des techniques de réduction de données. La seconde partie suit les mêmes étapes, avec l'ajout de la fonction perte centerloss aux classifieurs. La seconde partie contient aussi une tentative de modélisation en mixture de distribution Gaussienne des embeddings des CNNs.
- **Iteration 4 Detect anomaly with losses section clf with centerloss.ipynb :** notebook utilisant les classifieurs modélisés dans le notebook ci-dessus afin de détecter les anomalies, cette fois ci en utilisant la distance aux centroïdes calculées par fonction centerloss.