DataScientest • com

Rapport Technique d'évaluation

Détection de sons anormaux

Promotion: Juillet 2021

Participants :
NGUENGA Romuald
LECOINTRE Nicolas
BONTRON Francis

I. Contexte

D'un point de vue technique :

Il serait avantageux de pouvoir monitorer de manière continue des machines industrielles, et de pouvoir détecter lorsqu'elles sont défaillantes. En effet sur des chaînes de productions cela peut éviter des accidents et limiter les pertes dues à des interruptions de la production. L'approche étudiée dans ce rapport s'inspire de l'ouvrier mécanicien qui connaît parfaitement les machines sur lesquelles il travaille et qui est capable de détecter des pannes seulement à partir du bruit qu'elles émettent.

La détection des sons anormaux des machines industrielles rentre dans le cadre de la maintenance prédictive. L'objectif étant la collecte des données en temps réel au moyen de capteurs et l'analyse au travers des algorithmes de machine learning afin d'anticiper d'éventuels dysfonctionnements de la machine.

La détection par l'analyse du son est un moyen simple et peu onéreux de monitorer le bon fonctionnement d'une usine. Cela demande une certaine robustesse de la part du système, afin de ne pas interrompre la production pour des fausses alertes, ce qui se révélerait contre-productif. Mais avec un algorithme qui retourne peu de faux positifs on possède un moyen sûr et qui demande peu d'investissement pour assurer la sécurité de l'usine.

De manière plus générale, les outils de deep learning et de machine learning se placent comme des atouts pour l'industrie, mais peinent à percer. Selon le rapport du ministère de l'économie, Intelligence artificielle - État de l'art et perspectives pour la France, *"À l'heure actuelle, on assiste à un essor des data science utilisées à des fins d'optimisation de procédés. La dimension cognitive (apprentissage, décision) est encore peu présente dans ces développements. L'activité industrielle est soumise à des exigences fortes de fiabilité*". Or l'apport du sujet qui nous intéresse est justement de permettre un gain de fiabilité grâce à la dimension cognitive, ce qui faciliterait son acceptation.

D'un point de vue économique :

Selon l'Association Française des Ingénieurs et Responsables de Maintenance (AFIM), les coûts de la maintenance industrielle représentent à peu près 22 milliards d'euros par an en France. Ce montant élevé est dû principalement aux travaux de maintenance d'urgence qui nécessitent l'immobilisation des machines de production/fabrication.

Ainsi, pour réduire significativement ce chiffre, il est primordial que les entreprises, qu'elles soient de petites, moyennes ou grandes tailles, fassent appel à une nouvelle forme de maintenance, la maintenance prédictive. La force de la maintenance prédictive résidant dans l'anticipation des pannes, elle permettra aux entreprises d'éviter tout arrêt coûteux de la chaîne de production (à titre d'illustration, un arrêt de ligne de production chez Renault leur fait perdre environ 1 million d'euros par jour) et leur permettra de faire des économies non négligeables. Selon une étude du cabinet McKinsey, la maintenance prédictive permettra aux entreprises d'économiser 630 milliards de dollars d'ici 2025.

D'un point de vue scientifique :

L'analyse sonore à l'aide d'algorithmes de Deep Learning est une discipline encore très jeune. En 2018, Google parvient à isoler une voix au milieu de sons ambiants, ou encore à séparer les voix de deux personnes parlant en même temps pour n'entendre que la personne désirée. Avec la multiplication des "assistants virtuels" (*Alexa, Siri, Google Home, etc...*) la détection de voix est un enjeu important pour les développeurs de ces applications. Ces méthodes pourraient aussi se coupler aux systèmes de reconnaissance faciale pour ce qui est de la recherche en sécurité, ou encore à fournir des outils toujours plus performants dans le domaine de la musique, ou dans la gestion de l'acoustique.
De façon générale, il est important de comprendre les mécanismes entre la production et la compréhension d'un son.

I. Objectifs du projet

La problématique de ce projet consiste à détecter des anomalies sur des machines à partir d'un dataset ne contenant que des extraits en fonctionnement nominal. Sa résolution se décompose en trois grand objectifs :

- Traiter les extraits sonores pour les convertir en des données qu'un réseau de neurones peut utiliser.
- Développer plusieurs modèles avec différentes stratégies d'apprentissage afin d'évaluer la capacité à détecter des sons anormaux dans un set de test.
- Comparer les approches, aussi bien dans leurs résultats, la difficulté à les mettre en place, la rapidité d'exécution, etc...

Connaissances préalables :
Tous les membres du groupe avaient des connaissances de base en traitement du signal.

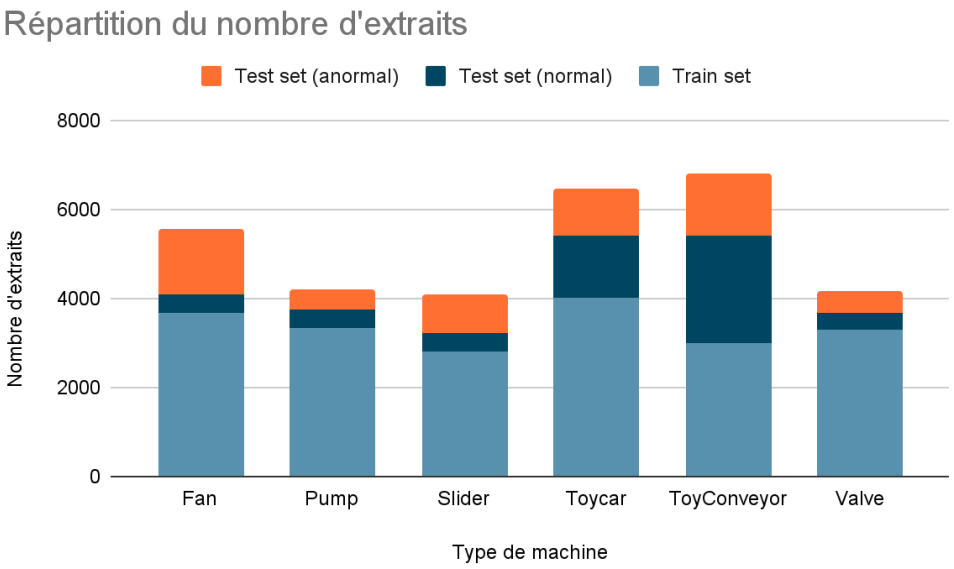
II. Data

A. Cadre

La problématique a été le sujet d'un challenge DCase en 2020 : *Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring*. Plusieurs équipes de data scientist ont soumis leurs propositions pour résoudre cette problématique, ce qui a été un point de départ pour envisager les différentes approches possibles et les performances qui étaient atteignables. Ce challenge étant public, les données étaient disponibles en accès libre.

Le jeu de données est constitué d'enregistrements sonores d'une durée de 10s et 11s provenant de 6 types de machines différentes (*Fan*, *Pump*, *Slider*, *Toy Car*, *Toy Conveyor* et *Valve*):

- 2 types de machines viennent du dataset ToyADMOS formé d'enregistrements sur des machines miniatures ("toy") à l'aide de 4 microphones.
- 4 autres types de machines viennent du dataset MIMII de machines réelles enregistrées par 8 microphones.



Pour réduire la taille du jeu de données, seul le premier canal de ces enregistrements est utilisé. Dans tous les enregistrements ont été gardés, de façon intentionnelle, les sons ambiants qui proviennent de différentes usines. Chaque type de machine est ensuite subdivisé selon un identifiant qui correspond à une machine spécifique. Ces machines ont ensuite été endommagées délibérément afin de produire les données sur les machines défaillantes. La variété des anomalies est telle qu'une liste exhaustive, sans connaître le détail de celles-ci, serait impossible à mettre en place.

Les données d'entraînements proviennent toutes d'enregistrements de machines en parfait état, alors que les données de validation sont plus ou moins équitablement réparties entre des machines défaillantes ou fonctionnelles, selon les types de machines. Le défi réside donc dans le fait que notre modèle de deep learning ne connaît pas à-priori les sons anormaux mais qu'il doit être capable de les identifier lors du test.

B. Pertinence

Un fichier audio n'est en réalité qu'une simple liste de valeurs numériques correspondant à l'intensité du son, ordonnées dans le temps pour former des sons différents. En l'état, peu d'informations peuvent en être récupérées.

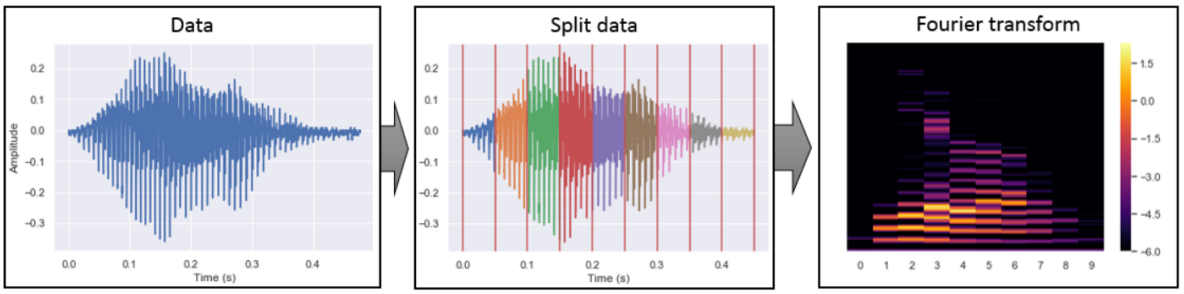
En analyse, une Transformation de Fourier (TF) permet de transformer une fonction intégrable dépendante du temps (un signal) en une fonction dépendante de la fréquence (un spectre).

$$F(\nu) = \int_{-\infty}^{\infty} f(t)\, e^{-i2\pi\nu t} dt$$

Toutes analyses sonores passent donc par l'étude des spectrogrammes produits par la transformée de Fourier du signal sonore.

Afin de garder une certaine dépendance temporelle, on utilise l'algorithme de Transformée de Fourier à Court Terme (*Short Time Fourier Transform*, *STFT*) :

Le signal est divisé en plusieurs parties (fenêtres) et une *Fast Fourier Transform* (*fft*, un algorithme de calcul de TF discrète) est appliquée sur chaque fenêtre. Cette technique nous permet de récupérer l'intensité de chaque fréquence dans le temps.

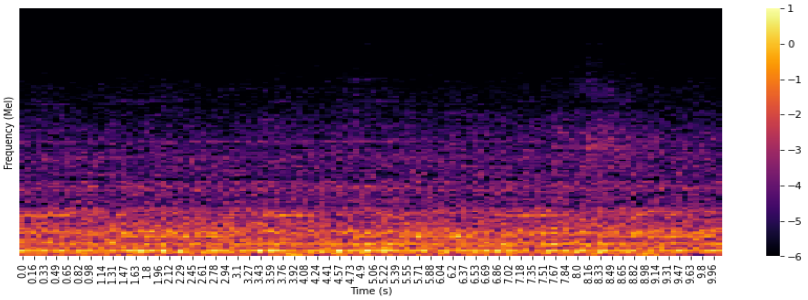


Les paramètres de la *STFT* sont les suivants:

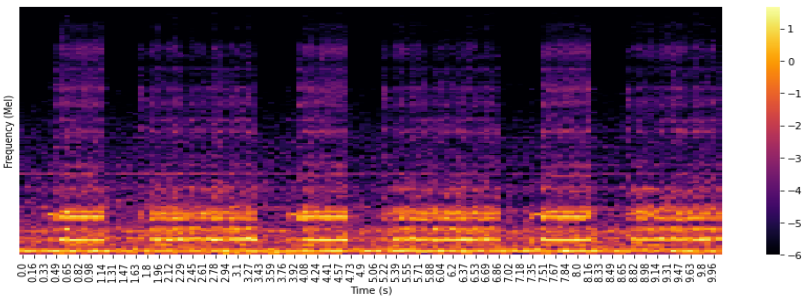
	Valeur	**Remarque**
Fréquence d'échantillonnage	16000 Hz	Déterminée par l'enregistrement
n_fft	1024	Taille d'une frame temporelle
frame_step	256	Chevauchement entre 2 frames
Intervalle de fréquences étudié	0-8000 Hz	Loi de Nyquist: moitié de la fréquence d'échantillonnage
num_mel_bins	128	Découpage fréquentiel

Nous travaillons donc avec des images dont l'intensité des pixels représente la puissance d'une bande fréquentielle pour une portion de temps donnée. Le problème se résume donc à devoir classifier des images représentant les spectrogrammes de fichiers audios selon deux catégories : machine normale ou anormale.

C. Analyse des spectrogrammes



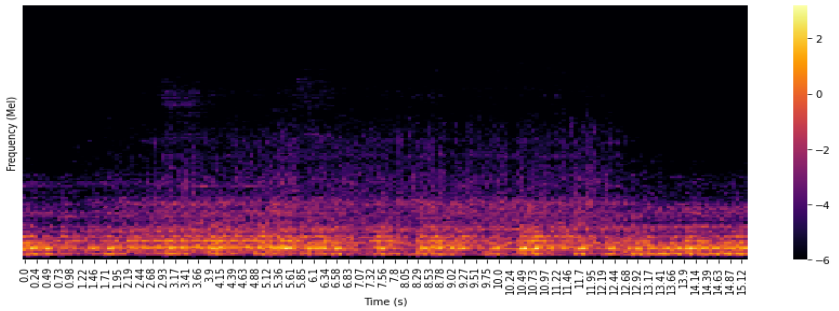
Extrait slider normal



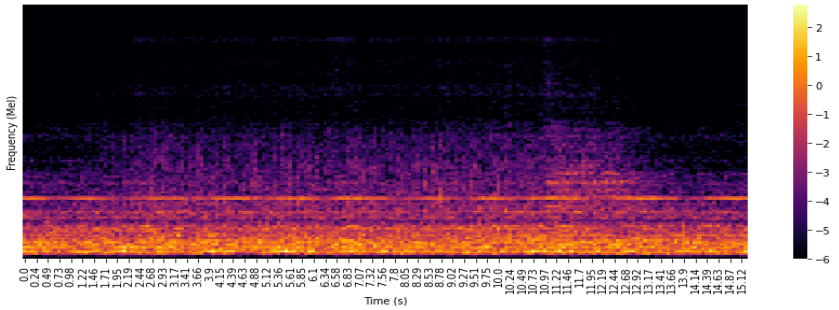
Extrait slider anormal

Sur ces extraits de slider, on peut aisément différencier l'extrait anormal de l'extrait normal, l'intensité du signal étant bien plus variable que dans un fonctionnement nominal. Par ailleurs, la distribution de fréquence se trouve changée, ce qui nous incite à s'intéresser aux variations à la fois des valeurs fréquentielles et temporelles.

En revanche si on trace les spectrogrammes de deux extraits "Toycar", on obtient les images suivantes:



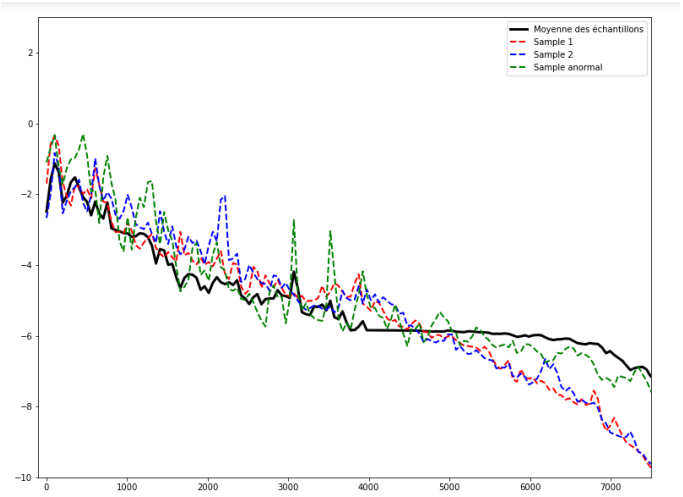
Extrait ToyCar Normal



Extrait ToyCar Défaillant

Dans ces cas là la distinction entre les images est bien plus subtile. Il semble qu'un nouveau pic de fréquence apparaît, mais cela nécessite de connaître avec précision les features sonores de la machine en bon état de marche.

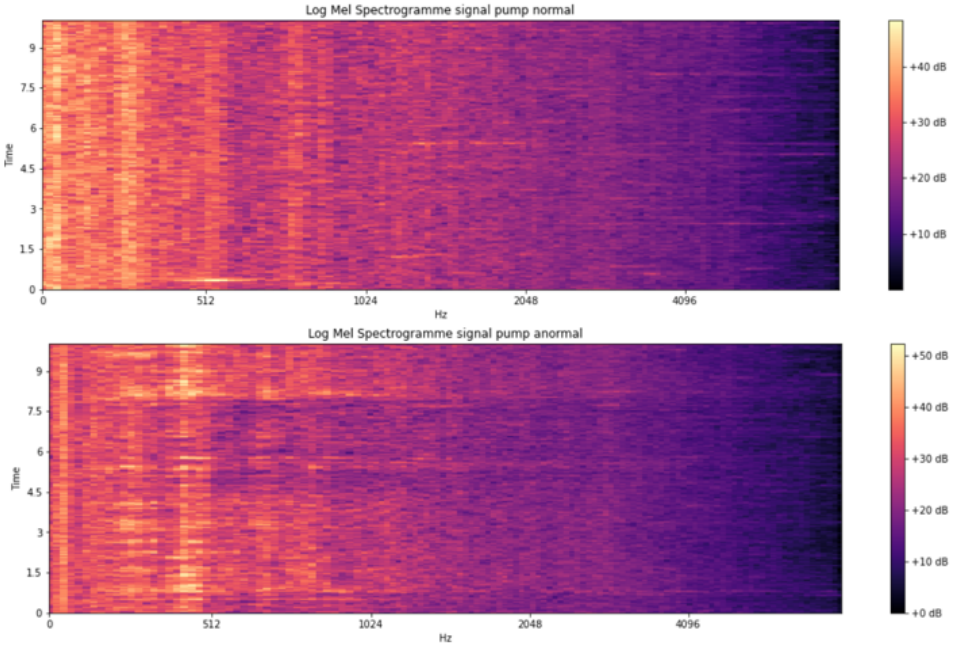
Si on trace la moyenne des intensités des fréquences pour étudier la distribution globale et de certains extraits normaux et anormaux, on constate qu'il n'y a pas de tendance globale qui permet de distinguer aisément chaque classe de son.



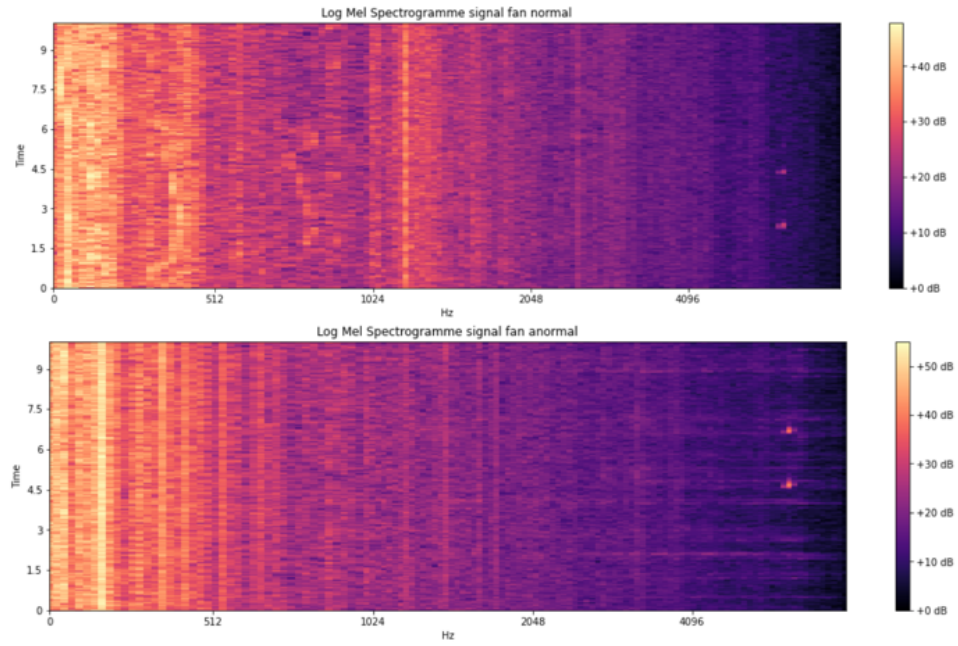
Comparaison des puissances moyennes de différents extraits

Cela confirme la nécessité de développer une méthode d'analyse plus complète, et le deep learning est un bon candidat étant donné la complexité des données et leur volume important.

Log Mel spectrogramme d'un signal normal et d'un signal anormal émis par la pompe



Log Mel spectrogramme d'un signal normal et d'un signal anormal émis par le ventilateur



En observant les spectrogrammes ci-dessus, nous constatons qu'une pompe défaillante ou un ventilateur défaillant émettent un signal sonore beaucoup plus bruité. Ces différences relevées aux niveaux des intensités sonores sur certaines machines donnent des pistes pour détecter les défaillances, mais ne seront pas nécessairement les métriques utilisées par les réseaux de neurones.

III. Projet

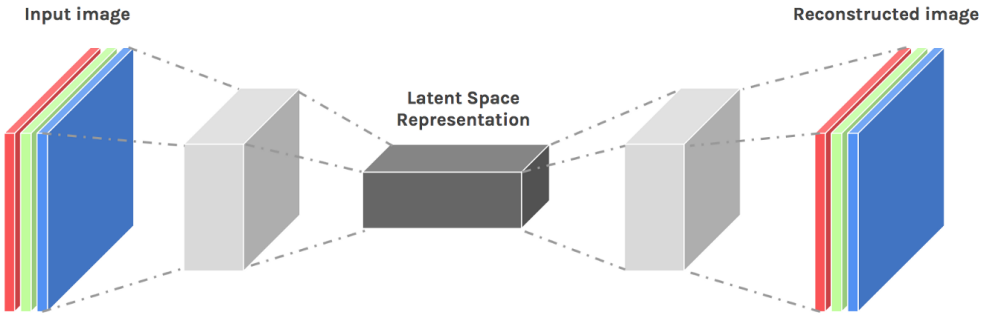
Notre projet de détection sonore de défaillance de machine s'apparente à un problème d'analyse d'images.

La grandeur discriminante pour juger de la performance de nos modèles sera la précision avec laquelle ils sont capables de différencier une machine fonctionnelle d'une machine défaillante.

En étudiant les soumissions au Challenge, deux grandes techniques semblent offrir de bons résultats, les auto-encodeurs et les classifieurs.

A. Les Auto-Encodeurs (AE) :

Un auto-encodeur est une architecture de réseaux de neurone particulière qui reconstruit les données d'entrée à sa sortie. L'intérêt d'une telle approche réside dans les couches intermédiaires dont les dimensions sont choisies de telle sorte qu'on extrait les caractéristiques ('features') de l'image d'entrée pour ensuite la reconstruire. Ainsi l'encodeur doit réduire progressivement la quantité de neurones à chaque couche, jusqu'à sa sortie encodée dont la dimension est souvent bien inférieure à l'entrée. Par la suite, le décodeur est construit en miroir par rapport à l'encodeur, pour remonter à la même dimension que l'entrée.

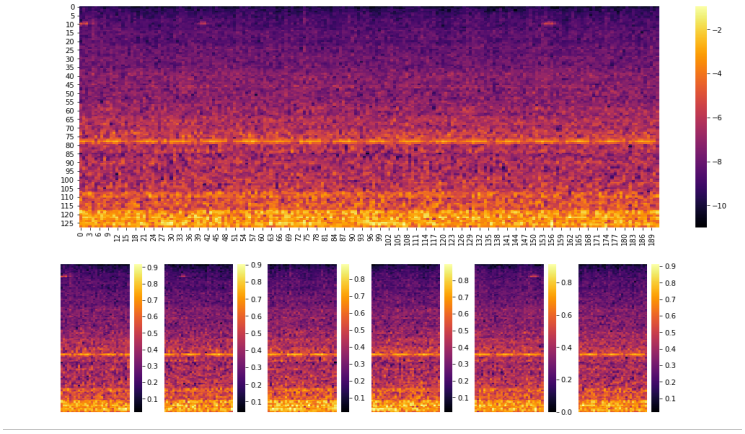


L'entraînement du modèle consiste à trouver les poids adaptés de telle sorte que la sortie soit la plus proche possible de l'entrée.

C'est une approche non supervisée, car le label des images n'a pas besoin d 'être connu. Lorsqu'on fournit au réseau entraîné sur des images normales une image anormale, il va avoir des difficultés pour la reconstruire à l'identique. Il suffit alors de mesurer l'erreur de reconstruction, qui est une estimation de l'écart entre l'image originale et l'image reconstruite, et de déterminer un seuil approprié afin de classer les images normales (faible erreur) et anormales (erreur supérieure au seuil)

D'un point de vue pratique, la première étape consiste donc à traiter les spectrogrammes de telle sorte que le set d'entraînement contient toutes les images correspondant au machines normales, classées en fonction de leur type et de leur ID. Dans une première approche nous avons séparé les ID pour entraîner l'auto-encodeur sur un ID bien spécifique afin d'en extraire les features.

Plusieurs dimensions d'images ont été testées, nous avons découpé les échantillons de 10s en 6 et 20 extraits, car nous ne savions pas *à priori* sur quel intervalle de temps les features seraient reconnues. (Cf *Conversion des fichier audio en images.ipynb)*. De plus, la réduction de la taille des images permet un traitement plus rapide par le réseau de neurones.



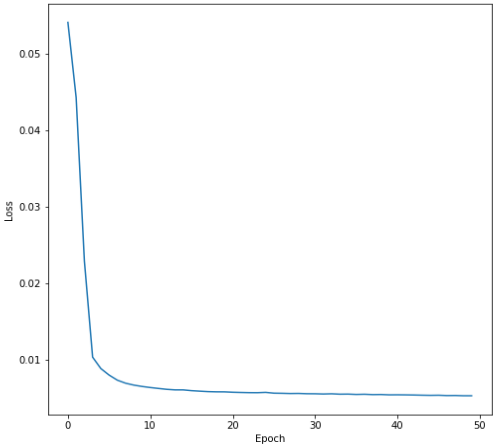
Les images sont ensuite normalisées et groupées en batchs, dont la taille doit être pertinente. En effet plusieurs itérations ont été menées et les batch de taille trop faibles ne sont pas pertinents car le réseau n'apprend pas les patterns spécifiques à la machine, il ne fait que recopier l'image d'entrée.

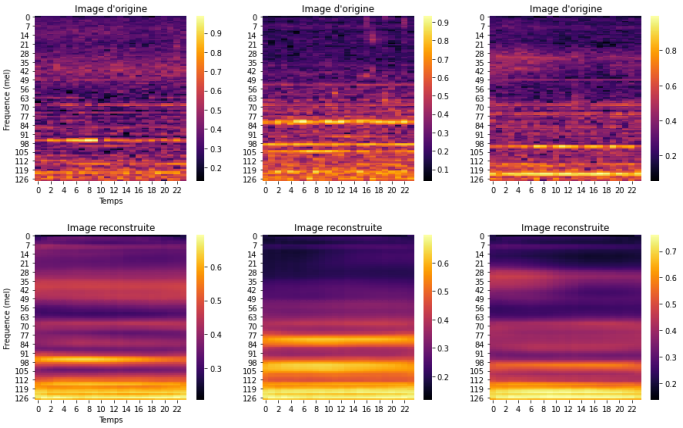
Par la suite l'auto-encodeur est construit avec les paramètres suivants (Cf *Modele autoencodeur v1)* . Le max pooling permet de réduire progressivement la dimension des images dans l'encodeur pour en extraire les features, puis le décodeur les ramène à la dimension initiale.

	Nom de la couche	Nombre de filtres	Taille du noyau	Activation
Encodeur	Convolution2D	64	5x5	Relu
	Max Pooling		2x2	
	Convolution2D	32	3x3	Relu
	Max Pooling		2x2	
	Convolution2D	16	3x3	Relu
	Max Pooling		2x2	
Décodeur	Convolution2D	16	3x3	Relu
	UpSampling		2x2	
	Convolution2D	32	3x3	Relu
	UpSampling		2x2	
	Convolution2D	64	5x5	Relu
	UpSampling		2x2	
	Convolution 2D	1	3x3	Sigmoid

Le modèle est initialisé avec des poids aléatoires ce qui permet de s'assurer que la descente de gradient converge vers le minimum global, et non pas vers une solution locale. La métrique utilisée pour mesurer la convergence du modèle est l'erreur de reconstruction, qui est la moyenne de l'écart au carré entre les pixels de l'image d'entrée et de sortie. C'est par ailleurs cette métrique qui est ensuite utilisée pour discriminer les extraits normaux et anormaux.

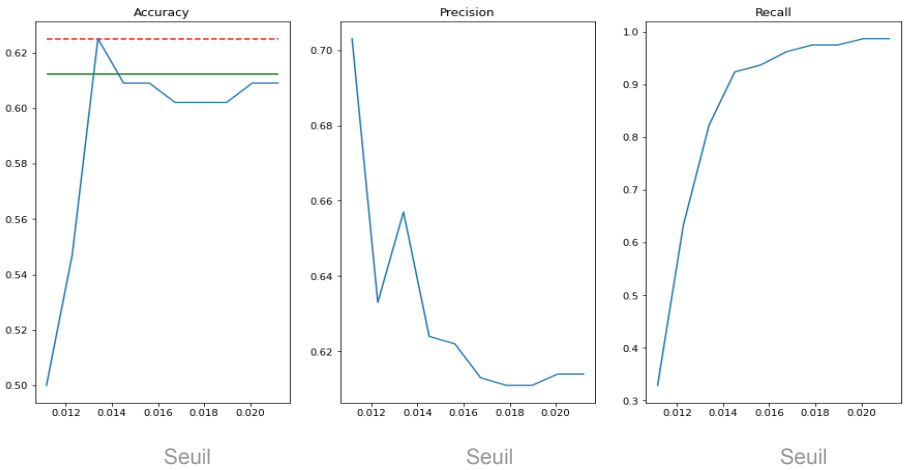
Une fois le modèle entraîné sur 50 époques avec un pas d'apprentissage choisi à 10^{-4},on vérifie la convergence de l'erreur et la reconstruction d'images prises au hasard :



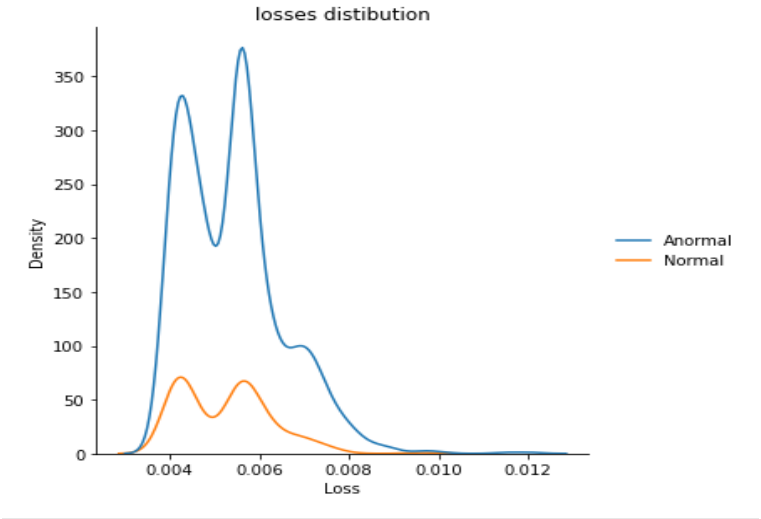


On constate que les images subissent un lissage qui est dû au fait que l'on crée des batchs de taille importante, dans le but de généraliser les extraits et de filtrer les bruit intempestifs.

Par la suite l'encodeur est utilisé sur le set d'images de test, en définissant un seuil d'erreur tel que les images dont l'erreur de reconstruction est supérieure à ce seuil sont classées comme anormales. La première approximation est de choisir ce seuil comme étant proche de la moyenne des erreurs sur le set d'entraînement, puis on le fait varier de quelques pourcents pour étudier l'évolution du classement des extraits. La métrique qui est principalement utilisée est l'accuracy, qui est comparée au seuil si tous les sons étaient classés dans la même catégorie. Par ailleurs la précision et le recall sont aussi calculés pour avoir des informations sur la manière dont le modèle classe les sons:



Le recall tend vers 1 à mesure que le seuil augmente, ce qui est cohérent avec le fait que de plus en plus d'extraits ont une erreur de reconstruction en dessous du seuil et sont donc classés comme normaux. Afin de visualiser pourquoi la position du seuil fait varier l'accuracy de cette manière, on trace un histogramme regroupant les erreurs de reconstructions, classé en fonction du label des extraits:



On constate alors que la distribution des pertes pour les sons anormaux est très similaire à celle des sons normaux, alors qu'on s'attendrait à un étalement des valeurs lors de la reconstruction de sons anormaux, car l'auto-encodeur a été entraîné seulement sur des extrait de machines en bon état de marche. La définition du seuil ne fait donc que passer des sons normaux en anormaux et inversement, mais ne discrimine pas clairement deux familles de sons après reconstruction. Nous avons essayé de faire varier divers paramètres pour pallier à ce problème :

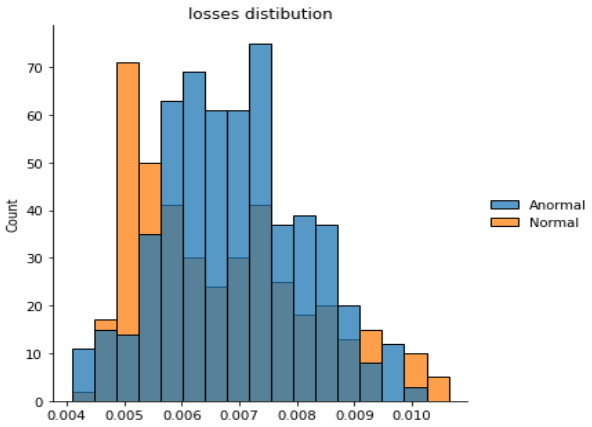
- variation de la taille des batchs pour mieux généraliser les caractéristiques de la machine
- modification des caractéristiques de l'auto-encodeur, en réduisant par exemple le nombre de couches pour que la réduction de dimension ne soit pas trop importante, ou augmentation du nombre de filtres pour capter plus de features
- mélange de différents ID de machine avec l'idée que les patterns spécifiques au type de machine devraient être séparés des patterns de l'environnement.

Cependant étant donné la volumétrie de données couplée à la quantité de machines, le temps nous a manqué pour trouver les combinaisons de paramètres les plus adaptées pour chaque machine. Les résultats présentés ici ne reflètent donc pas les meilleurs résultats qui puissent être obtenus grâce à notre modèle, mais démontrent la nécessité d'essais plus approfondis avec une plus grande puissance de calcul .

On remarque cependant une tendance globale d'une machine à l'autre: l'erreur de reconstruction des extraits anormaux n'est jamais clairement différentiable de celle des sons normaux. Les résultats obtenus pour les différentes machines sont présentés dans le tableau suivant, l'accuracy étant comparée à sa valeur si tous les extraits étaient considérés normaux.

	Accuracy / All Normal	Précision	Recall
Valve	0.3 / 0.43	0.35	0.8
Toycar	0.58 / 0.55	0.56	0.95
ToyConveyor	0.64 / 0.62	0.67	0.9
Fan	0.45 / 0.18	0.17	0.57
Pump	0.67 / 0.55	0.7	0.4

Un recall élevé assure qu'on ne classe pas trop de sons normaux en anormaux, ce qui se traduirait dans notre problématique par des interruptions non voulues des chaînes de production. La précision mesure la bonne détection des anomalies. Ainsi l'analyse de la pompe donne des résultats intéressants, les anomalies étant relativement bien détectées, mais un recall faible. La distribution des pertes est légèrement dispersée entre les 2 labels, ce qui permet de capter une bonne partie des défaillances sans trop de faux négatifs.



Pour le reste des machines la définition du seuil revient seulement à prédire quasiment aléatoirement les labels, car les reconstructions sont trop similaires.

Le problème semble provenir de la nature même des spectrogrammes, qui ne sont pas des images telles que celles traitées habituellement par les auto-encodeurs. Tout d'abord la position des features sur l'image est importante, car un déplacement vertical de pixels (qui est donc une augmentation de la fréquence) ne sera pas capté par la convolution, mais peut en réalité correspondre à une défaillance. Par ailleurs la nature temporelle du signal n'est pas prise en compte dans cet auto-encodeur "basique", or certaines machines ont un comportement cyclique qu'il serait pertinent de pouvoir capter. En effet les défaillances peuvent prendre la forme d'une rupture de ce caractère cyclique, le meilleur exemple étant le fan qui tourne régulièrement en fonctionnement normal, et présente des pics sonores aléatoires lorsqu'il est défaillant.

B. Les Classifieurs

Dans un problème de classification classique, nous entraînons les modèles sur un jeu de données contenant toutes les classes à prédire. Dans notre cas, seule la classe “Normal” est présente dans le dataset d’entraînement.

Une astuce pour contourner le problème consiste, non pas à prédire si le spectrogramme en entrée appartient à une machine est “Normal” ou “Anormal”, mais plutôt de prédire à quel type de machine appartient cet enregistrement. Pour cela, on entraîne le modèle sur un ensemble de machines en spécifiant quelle machine appartient à la classe négative (*expl* : *Pump*) et lesquelles appartiennent à la classe positive (*expl* : *Fan*, *Valve* et *Slider*). On demande ensuite à notre modèle d’évaluer les données provenant de la machine que l’on souhaite tester (ici *Pump*) en définissant la classe négative pour les machines “Normale” et la classe positive pour les machines “Anormale”. Le modèle étant entraîné à reconnaître une machine en particulier, les sons qui ne sont pas reconnus comme provenant de cette machine seront considérés anormaux.

Comme nous transformons notre problème en reconnaissance d’image, nous utilisons un réseau de neurones convolutif (*Convolutional Neural Network* ou *CNN*). Pour rappel, un CNN reçoit des images en entrée, détecte les features de chacunes d’entre elles, puis entraîne un classifieur dessus.

i. Dataset

Pour notre modèle de classification nous considérons comme sons normaux les données sonores issues du fichier pump/train ⇒ label 0.

En parallèle les données sonores issues des autres datasets d'entraînement (fan, valve et slider) sont considérées comme sons anormaux ⇒ label 1.

Ainsi, notre dataset d’entraînement est constitué de 25% de données sonores normales et de 75% de données sonores anormales.

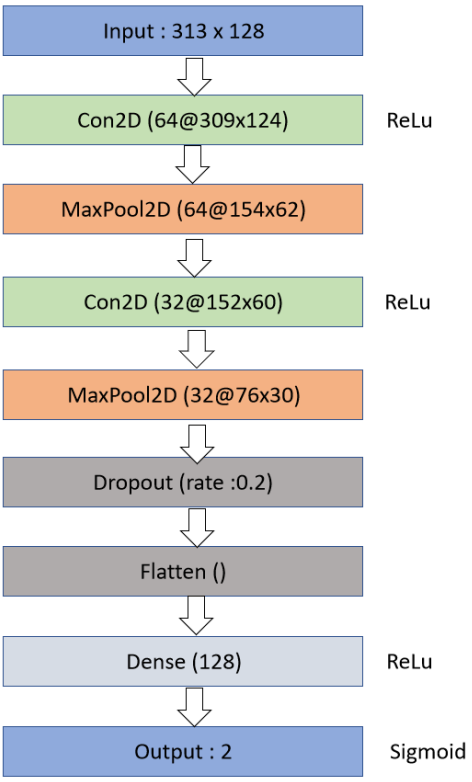
Le dataset de test est composé de 47% des données sonores de pompes normales et de 53% de données sonores de pompes anormales.

ii. Pré-processing des données

Avant d'entraîner notre modèle de classification, une étape de pré-processing est nécessaire. Pour ce faire, nous avons utilisé la librairie tensorflow.data.Dataset qui nous permet de transformer nos données sonores en spectrogrammes.

iii. Modèle classification

Pour la partie classification, nous nous sommes inspirés de l’architecture LeNet en augmentant le nombre de filtres. En effet, l’architecture LeNet telle qu’elle est connue n’arrivait pas à extraire assez de features des différentes images pour pouvoir ensuite les classer.



iv. Résultats :

Plusieurs itérations ont été réalisées sur notre modèle afin de déterminer les paramètres optimaux.

Dans un premier temps, il nous a fallu trouver un compromis entre la dimension de l'image en entrée du modèle qui garderait un maximum d'informations du signal temporel et le temps d'entraînement. Cette étape a été réalisée en faisant des itérations sur les paramètres de la fonction tfio.audio.spectrogram qui convertir un signal audio en spectrogramme. Ainsi, pour notre modèle, les paramétre retenus pour la fonction tfio.audio.spectrogram sont :

- nfft (nombre d'échantillon dans un frame) = 4096;
- window (Taille du frame temporelle) = 4096;
- stride (Chevauchement entre deux frame) = 512

Nous nous sommes également rendu compte que la taille des batchs était assez significative pour notre modèle. A titre d'illustration, l'utilisation d'un batch de taille 64 nous a fait gagner 6% sur la précision du modèle.

Une fois les paramètres de notre modèle définis, nous l'avons entraîné avec plusieurs configurations du dataset d'entraînement.

En effet, les différences dans les enregistrements de deux machines différentes sont souvent plus importantes que les différences entre un son normal et un son anormal pour

une même machine. Pour contourner ce problème, on essaye d'entraîner les modèles sur différentes instances (ou "id") d'une même machine, très similaires, et de chercher à détecter parmi quelques instances seulement les sons anormaux. Entre autres, sur la machine pompe, nous avons quatre instances.
Dans un premier temps, nous avons considéré uniquement la pompe id = 00 comme son normal et les autres "id" de la pompe comme sons anormaux.
Nous constatons que l'accuracy de notre modèle augmente au fur et à mesure que nous considérons de plus en plus de "id" comme sons normaux. Ce qui pourrait s'expliquer par le fait que notre réseau dispose de plus en plus de données sonores normales pour s'entraîner (cf. tableau de synthèse ci-dessous).

Différentes configurations du dataset d'entraînement	Accuracy (%)
config_1 : Train/pump/Id = {00} => label 0	55
config_2 : Train/pump/Id = {00, 02 } => label 0	57
config_3 : Train/pump/Id = {00, 02, 06 } => label 0	62
config_4 : Train/pump/Id = {00, 02, 04, 06 } => label 0	71

Ainsi, nous obtenons une meilleure précision en considérant comme sons normaux tous les "id" de la pompes contenues dans le set d'entraînement.

En considérant donc la configuration n°4, nous avons comme résultat :

v. Matrice de confusion :

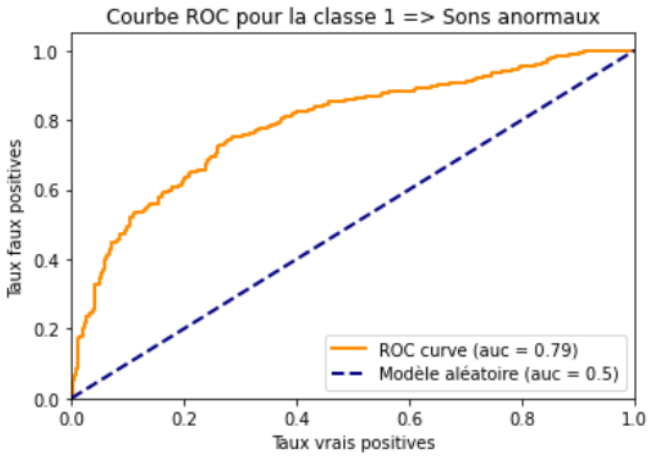
valeur prédite / valeur réélle	0	1
0	332	68
1	176	280

⇒ Le modèle a du mal à classer les données sonores anormales de la pompe. En effet, sur 456 données sonores anormales seules 61% sont prédites par le modèle, ce qui fait qu'il reste 38% de sons anormaux qui ne sont pas détectés.

vi. Rapport de classification

	precision	recall	f1-score	support
0	0.65	0.83	0.73	400
1	0.80	0.61	0.70	456
accuracy			0.71	856
macro avg	0.73	0.72	0.71	856
weighted avg	0.73	0.71	0.71	856

vii. courbe ROC (Receiver Operating Characteristic) pour la classe 1 (sons anormaux)



⇒ L'aire sous la courbe AUC (Area Under the Curve) n'est que 0.79 et la courbe ROC reste encore bien éloignée du point (0,1). Notre modèle n'est pas encore assez performant pour bien classifier nos données sonores.

En nous basant sur les paramètres qui nous ont permis de classifier les pompes, nous avons construit des modèles pour les autres machines (fan, slider, valve).
Ci-dessus un tableau récapitulatif qui donne les "AUC" sur les différentes machines

	Pump	Fan	Slider	Valve
AUC	0,79	0,70	0,55	0,56

⇒ Le comportement du modèle pour classer le slider et la valve (AUC environ 0,55) n'est pas très différent du comportement d'un modèle aléatoire (AUC = 0,5). Il faudrait reconstruire un tout autre modèle avec des nouveaux paramètres afin de mieux classer les données sonores des sliders et des valves.

IV. Difficultés rencontrées

Une des premières difficultés rencontrée lors du projet fut la taille du jeu de données, qui regroupe un grand nombre d'échantillons à traiter. C'est une caractéristique avantageuse du point de vue de l'apprentissage des réseaux de neurone, le risque d'overfitting étant réduit, mais cela nécessite une bonne puissance de calcul. Dans un premier temps la vérification du bon fonctionnement des codes a été réalisée sur des jeux d'entraînement réduits, tirés aléatoirement parmi les jeux complets.

D’un point de vue plus technique, nous n’avions pas initialement les connaissances suffisantes en réseau de neurones et en code pour mettre immédiatement en place les approches envisagées, nous avons dû d'abord nous former sur des cours de la plateforme qui étaient planifiés pour la fin de la formation. Cela a retardé le démarrage de la partie code, et a constitué un défi vis-à-vis de la gestion du temps. En effet, il nous fallait progresser suffisamment rapidement sur les cours pour atteindre les modules pertinents pour notre étude, sans bâcler les certifications obligatoires à valider simultanément. Par ailleurs, une partie des connaissances (notamment sur les auto-encodeurs) n’était pas couverte par les cours, ce qui a nécessité des recherches personnelles approfondies qui requéraient elles aussi une bonne base théorique pour comprendre comment les déployer. Avec plus de temps nous aurions pu mettre en place des approches plus complexes et plus adaptées à la résolution de cette problématique.

Pour ce qui est de la problématique en elle-même, la difficulté résidait dans la subtilité nécessaire pour discriminer les cas normaux et anormaux. Les enregistrements sont bruités et captés dans des environnements différents, parfois avec des sons ambiants liés au fonctionnement de l’usine, et les défaillances peuvent prendre des formes variables. Ainsi, les défaillances sont facilement assimilables à un bruit ambiant et inversement, complexifiant davantage la tâche.

V. Bilan & Suite du projet

Pour rappel, pour résoudre notre problématique qui était de détecter des machines défaillantes en n’ayant entraîné nos modèles que sur des données normales, les différents objectifs du projet étaient les suivants:

- Objectif 1 - Traitement des signaux sonores
- Objectif 2 - Déploiement d’algorithmes de deep learning
- Objectif 3 - Comparaison des approches et des résultats obtenus

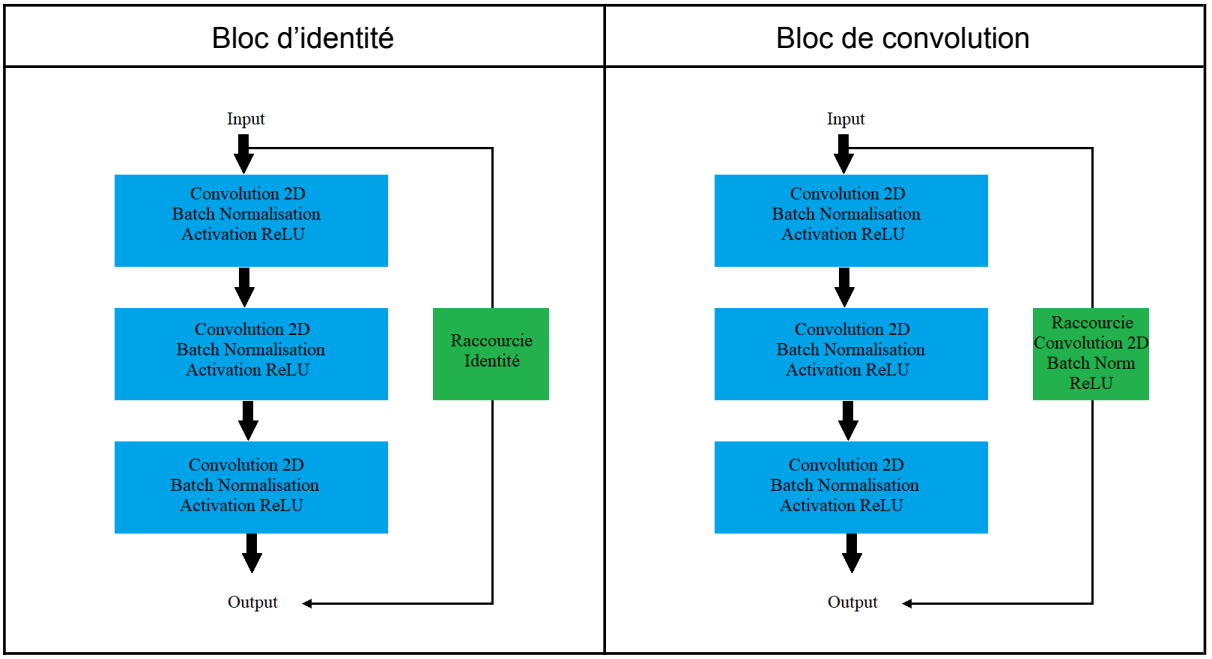
Les deux premiers objectifs ont été complétés, les extraits audio ayant été traités pour être analysés, et deux approches de deep learning ayant été développées pour résoudre la problématique de détection d’anomalies. Cependant les résultats apportés par les deux méthodes sont mitigés, car le taux de détection d’anomalies n’est pas assez satisfaisant pour envisager un déploiement. Seule l’analyse de la “Pump” donne des résultats intéressants avec les deux réseaux. L’objectif 3 n’est donc que partiellement rempli, et nécessiterait de revoir les approches que nous avons envisagées.

Les limites des modèles que nous avons mis en place nous permettent d’envisager des méthodes plus approfondies :

- Pour l’auto-encodeur, une prise en compte de la temporalité pourrait apporter un vrai plus au modèle, notamment en intégrant des couches avec une mémoire (LSTM) afin de détecter les phénomènes cycliques. Par ailleurs, certaines soumissions du challenge ont démontré que l’utilisation de masques sur les données d’entrée permet

aussi d'améliorer les performances de l'auto-encodeur, qui doit alors reconstruire les spectrogrammes en ayant une partie de l'information cachée.

- Pour la classification : De façon générale, il semble qu'augmenter le nombre de couches d'un réseau permet de réduire l'erreur, au prix de performances de calcul. Or, à partir d'un moment, la précision semble se dégrader. Pour pallier ce problème, l'architecture *ResNet* (*Residual Network*) utilise des sauts de connections (ou raccourcis) qui peuvent être ou des fonctions identités, ou des fonctions de convolutions. Ces raccourcis sont, dans un premiers temps, instanciés au sein d'un bloc (Bloc d'identité ou de convolution), puis ces blocs sont ajoutés au modèle de la même façon que les autres couches. Dans le pire des cas, le réseau n'apprend rien de plus que si ces raccourcies n'existaient pas, sinon, ils permettent une amélioration de l'apprentissage en permettant aux couches en profondeur d'apprendre sur des données moins transformées par les couches supérieures. Des essais de classification ont été réalisés, mais sans résultats pertinents, ou avec des performances moindre qu'avec le réseau LeNet : sur les différents essais, le modèle semble classer l'ensemble des données tests dans la même classe, ne faisant aucune distinction entre les machines normales et anormales.



VI. Bibliographie

Impact économique:

https://fr.rs-online.com/web/generalDisplay.html?id=discovery-maintenance/comment-reduire-le-cout-de-la-maintenance-dans-l-industrie

https://www.sicara.fr/parlons-data/2018-03-21-maintenance-predictive

Dcase Challenge

Description and discussion on DCase2020 challenge Task2: Unsupervised anomalous sound detection for machine condition monitoring

Yuma Koizumi1, Yohei Kawaguchi2, Keisuke Imoto3, Toshiki Nakamura2, Yuki Nikaido2, Ryo Tanabe2,Harsh Purohit2, Kaori Suefusa2, Takashi Endo2, Masahiro Yasuda1, Noboru Harada1,

Auto Encodeurs:

https://towardsdatascience.com/implementing-an-autoencoder-in-tensorflow-2-0-5e86126e9f7

https://medium.com/analytics-vidhya/autoencoders-with-tensorflow-2f0a7315d161

https://www.tensorflow.org/tutorials/generative/autoencoder

Self_supervised classification for detecting anomalous sounds

Ritwik Giri, Srikanth V. Tenneti, Fangzhou Cheng, Karim Helwani,Umut Isik, Arvindh Krishnaswamy

ResNet:

https://ichi.pro/fr/cnn-architecture-comment-fonctionne-resnet-et-pourquoi-30895772711476

https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d#e276

https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

VII. Annexes

- Modèle AE

Data Viz.ipynb: Tracé des spectrogrammes de 2 machines et visualisation de différentes caractéristiques.

Conversion des fichier audio en images.ipynb: Chargement du fichier contenant les extraits audio, en spécifiant le chemin d'accès. Création et sauvegarde d'un DataFrame contenant les caractéristiques des extraits (Machine, Id, Labels). Calcul des spectrogrammes log-mel associés, qui sont ensuite sauvegardés sous formes d'images normalisées dans un nouveau dossier.

Modèle autoencodeur.ipynb: Importation des images d'une machine et de 1 ou 2 ID, et mise en forme des tenseurs d'entraînement et de test. Définition du modèle d'auto-encodeur, ainsi que des fonctions utiles au calcul de perte, à la descente de gradient, et aux prédictions. Entraînement du modèle, tracé des images reconstruites, des prédictions et des métriques correspondant à l'ensemble de test.

- Modèle classifieur

Chargement_fichier_dataset.ipynb : Permet de créer des dataframes contenant les chemins d'accès aux fichiers audio sur répertoire et de les labelliser.

Data_Viz.ipynb : Permet de tracer des spectrogrammes et de visualiser certaines caractéristiques.

model_classification_pump_V3.ipynb, model_classification_fan.ipynb, model_classification_slider_V1.ipynb, model_classification_valve.ipynb: Définition d'une fonction permettant de convertir les données sonores en spectrogramme. Ensuite cette fonction est utilisée dans une pipeline de tensorflow.data.Dataset. Définition du modèle de classification, entraînement et affichage des résultats.

- Modèle ResNet :

model_resnet.ipynb : Convertit les fichiers audios en spectrogramme, définit un modèle ResNet et présente les premières estimations.