rectangle blanc pour le texte sur la couverture

|  |
| --- |
| séparateur de texte |
|  |
| Promotion Datascientist Mai 2021  séparateur de texte  Astrid LANCREY  Corentin DESMETTRE  Yossef ABOUKRAT  Youcef MADJIDI |

rectangle coloré

# Sommaire

Table des matières

[**Sommaire 2**](#_heading=h.gjdgxs)

[**Contexte 3**](#_heading=h.30j0zll)

[**Problématique 3**](#_heading=h.1fob9te)

[**Objectif 3**](#_heading=h.3rdcrjn)

[**Data 3**](#_heading=h.3znysh7)

[**Cadre 3**](#_heading=h.26in1rg)

[**Projet 3**](#_heading=h.2et92p0)

[**Classification du problème 3**](#_heading=h.tyjcwt)

[**Choix du modèle & Optimisation 4**](#_heading=h.3dy6vkm)

[**Difficultés rencontrées lors du projet 4**](#_heading=h.1t3h5sf)

[**Bilan et suite du projet 4**](#_heading=h.4d34og8)

[**Titre de l’en-tête 5**](#_heading=h.2s8eyo1)

[**Texte du sous-titre ici 5**](#_heading=h.17dp8vu)

# Contexte

La robotisation et l’automatisation des machines et des biens est un enjeu du 21-ème siècle. A cette fin, de plus en plus d’industries investissent pour rendre totalement autonome des parties de leur process ou des tâches qui nécessitent habituellement la présence constante d’un opérateur. Les exemples les plus représentatifs dans l’industrie sont les robots de manutention et les chaînes d’assemblage automatiques. Concernant le quotidien, la course à la voiture autonome et aux drones de surveillance / livraison en sont des exemples les plus visibles.

La problématique associée à ces enjeux est l’auto-maintenance des machines ; tâche de surveillance réalisée dans le passé par l’opérateur, son absence induit un contrôle moins fréquent des paramètres et donc un risque de panne plus élevé. Dans cet objectif, l’enjeu du projet est de pouvoir détecter une anomalie dans une machine grâce à sa signature audio

## Problématique et objectif

Le principal défi de cette tâche est de détecter des sons anormaux qui sont par définition inconnus. En effet, dans les usines, les sons anormaux réels sont très diversifiés et nous ne pouvons avoir une liste exhaustive de ceux-ci. Cela signifie que nous devons détecter des sons anormaux inconnus qui n'ont pas été observés dans les données d'entraînement fournies. Aussi, ce problème ne peut être vue comme un simple problème de classification a deux classes (normal/anormal).

Nous tenterons donc au travers de différentes approche et model de détecter une anomalie dans une machine grâce a sa signature audio. Le livrable principal de ce projet se constituera d’un algorithme de contrôle et d’alerte implémentable dans l’appareil industriel et pourquoi pas par la suite l’intégrer dans un dispositif (type Raspberry Pi équipé d’un micro) configuré pour une surveillance en temps réel de la machine.

# Data

Le jeu de donnée mis a disposition est celui disponible sur le site [KAGGLE](https://www.kaggle.com/daisukelab/dc2020task2) et représente 9,42 Go de fichier son au format WAV. On y retrouve des parties de différentes DataBase tel que ToyADMOS et de l'ensemble de données MIMII. Constituées des sons de fonctionnement normaux/anormaux de six types de machines jouets/réelles. Chaque enregistrement est un audio à canal unique (approximativement) d'une longueur de 10 secondes qui comprend à la fois le son de fonctionnement de la machine cible et le bruit ambiant. Les six types de machines jouet/réelles suivants sont utilisés dans cette tâche :

## DataViz

# Projet

## Classification du problème

Nous avons ici à faire à une problématique de détection d’anomalies qui diffère d’un problème de classification “classique” de par la sous-représentation d’une des deux classes de données à identifier: à savoir les fichiers son correspondant à des sons anormaux émis par une machine.

## Choix du modèle & Optimisation

Pour générer un modèle qui permettrait de de détecter les anomalies nous avons exploré quatre approches distinctes:

1- Autoencoder

En nous inspirant de la “baseline” qui est le modèle proposé par les coordinateurs du challenge DCASE2020, nous avons tout d’abord mis au point un Autoencoder que nous avons entraîné à reconstituer les spectrogrammes logMel issus des fichiers son de type normal. Le but étant d’obtenir des reconstitutions assez fidèles pour que la MSE calculée à partir des pixels du spectrogramme initial et après reconstitution par notre AE augmente lorsque les reconstitutions sont prédites sur des fichiers anormaux. Nous avons testé en parallèle un AE constitué de couches de convolution 2D et un AE similaire à celui utilisé dans la baseline constitué de couches de type dense. Nous présenterons dans une première sous-partie les distributions des Mean Square Error(MSE) obtenues sur l’ensemble des fichiers de type normal et anormal au fil des optimisations testées sur un AE de convolution. Dans un second temps nous présenterons les résultats obtenus sur un AE dense.

1-a Autoencoder de convolution

2/ Modèle de classification du type de son normal vs anormal de type hybride

Les résultats de reconstitution obtenus par les Auto-encoder ne permettant pas de classifier si le son d’entrée était normal ou anormal nous avons décider de valider la faisabilité de la classification en remplaçant la partie décoder (convolution, upsampling) par des couches de classification dense.

La démarche étant une étude de faisabilité, nous avons entraîné le modèle sur les données du train et du test, en se servant de générateur de données afin d’augmenter la quantité initiale d’anomalies afin d’atteindre au moins 30% dans l’échantillon final. D’un point de vue utilisation finale cette démarche serait équivalente à entraîner un auto-encoder sur une machine saine lors de la conception, puis d'entraîner le modèle décisionnel sur différentes anomalies de la même machine pour enfin ne ré-entraîner que la partie auto-encoder sur la machine cliente saine finale.

Nous avons fait varier plusieurs paramètres :

* 1 à 3 couches Denses cachées avant la couche de sortie
* 64 à 1024 Neuronnes par couche
* Avec et Sans Dropout
* Différentes activation finale et par couche (softmax, sigmoid, tanh, relu,leakyrelu)

Layer (type) Output Shape Param #

=================================================================

conv2d\_11 (Conv2D) (None, 256, 256, 64) 640

max\_pooling2d\_5 (MaxPooling2 (None, 128, 128, 64) 0

conv2d\_12 (Conv2D) (None, 128, 128, 32) 18464

max\_pooling2d\_6 (MaxPooling2 (None, 64, 64, 32) 0

conv2d\_13 (Conv2D) (None, 64, 64, 16) 4624

max\_pooling2d\_7 (MaxPooling2 (None, 32, 32, 16) 0

conv2d\_14 (Conv2D) (None, 32, 32, 8) 1160

max\_pooling2d\_8 (MaxPooling2 (None, 16, 16, 8) 0

conv2d\_15 (Conv2D) (None, 16, 16, 4) 292

max\_pooling2d\_9 (MaxPooling2 (None, 8, 8, 4) 0

flatten (Flatten) (None, 256) 0

dense (Dense) (None, 1024) 263168

dropout (Dropout) (None, 1024) 0

leaky\_re\_lu (LeakyReLU) (None, 1024) 0

dense\_1 (Dense) (None, 1024) 1049600

dropout\_1 (Dropout) (None, 1024) 0

leaky\_re\_lu\_1 (LeakyReLU) (None, 1024) 0

dense\_2 (Dense) (None, 1) 1025

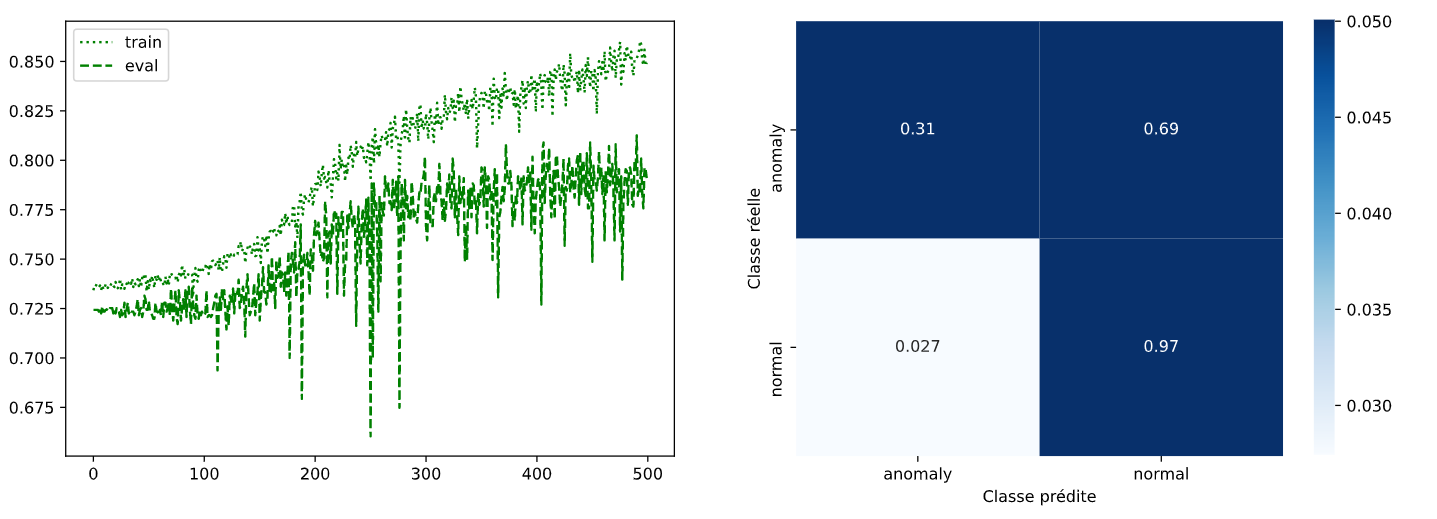
=================================================================

Total params: 1,338,973

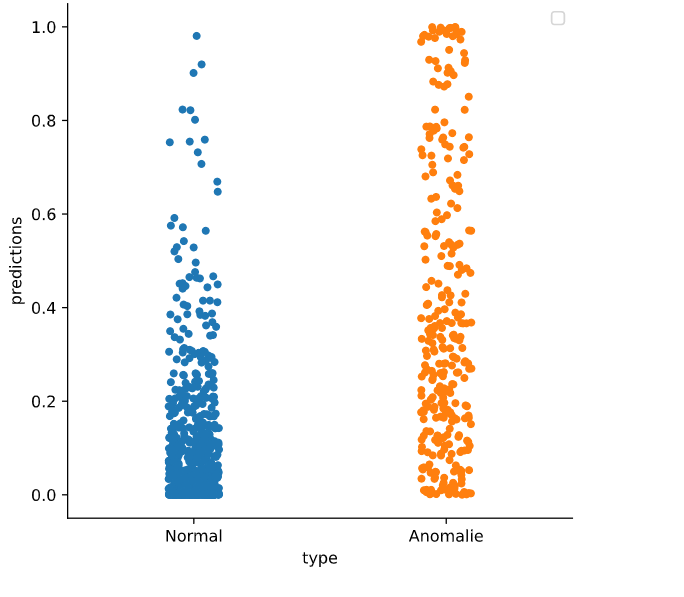
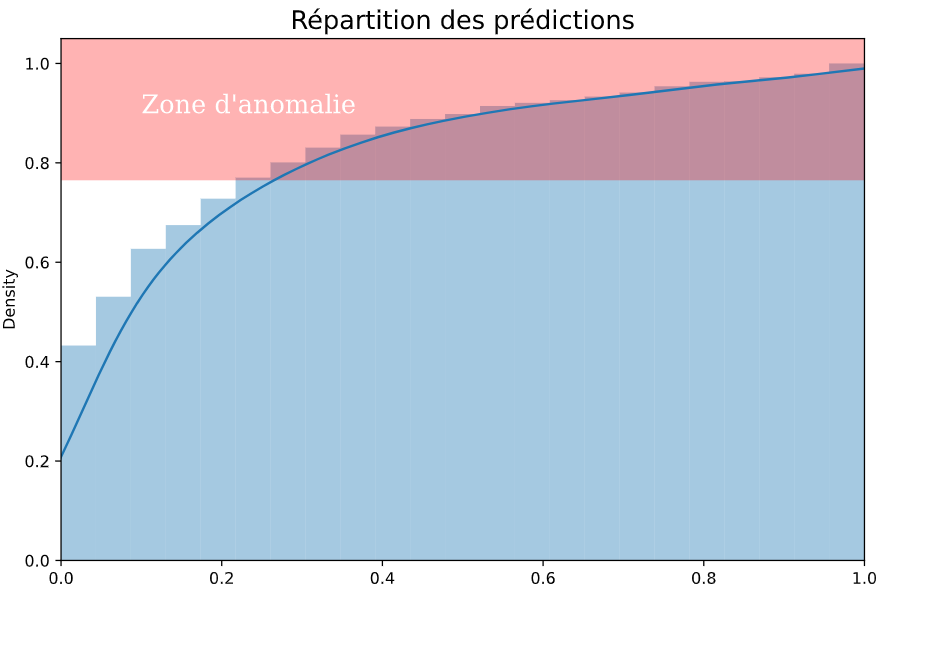
Trainable params: 1,313,793

Non-trainable params: 25,180

Entrainement :



Résultats :



En regardant la répartition de probabilité par type (figure de droite) nous pouvons voir qu’un réglage de la sensibilité pourrait être à 0.4 au lieu de 0.5. Et le faire nous permet en effet de détecter 7% d’anomalies supplémentaires mais avec 2 fois plus de fausses détections (5% au lieu de 2.5%).

Conclusion :

Ce modèle valide la capacité de prédiction normal/anormal sur notre jeux de donnée et avec nos outils de traitement.

3/ Modèle de classification des ID de type hybride

Ne pouvant se servir des données de test afin de pouvoir entraîner les modèles dans le projet, nous avons du trouver une autre solution ne mettant en oeuvre que les données d'entraînement pour entraîner le modèle.

L’idée suivie est donc d'entraîner un modèle à détecter les IDs des machines sur le train, et de catégoriser un échec comme une détection d’anomalie sur le test. L’objectif est donc d’avoir un modèle suffisamment entraîné pour détecter à coup sur l’ID machine lorsque le son est normal, mais suffisamment sensible pour qu’une anomalie dans ce son provoque une forte disparité de classement.

Nous avons donc dans un premier temps réaliser une classification ascendante hiérarchique sur les données train et test prédites par la couche finale de la partie encoder (vecteur de compression). Les résultats obtenus à partir des prédictions sur le train donc sans anomalies sont présentés à gauche. Ceux obtenus à partir des prédictions sur le jeu de donées test contenant des anomalies sont présentés à droite:

### Nous pouvons identifier quatre clusters dont on peut supposer qu’ils correspondent aux quatre ID de la machine fan. Cela nous prouve la faisabilité de détecter les classes des machines même sur des données fortement compressées.

4/ Modèle de classification des ID de type hybride

### Une fois ce contrôle effectué, nous avons ensuite ajouté des couches dense derrière la partie le vecteur de compression de la partie Encoder, la dernière comportant 4 neurones et une fonction d’activation softmax dans le but de classifier nos ID à partir des features du vecteur de compression de notre AE. Nous avons dans un premier temps testé avec autant de neuronnes sur les couches Denses que de variables dans la couche Flatten.

Nous avons fait varier plusieurs paramètres :

* 1 à 3 couches Denses cachées avant la couche de sortie
* 64 à 1024 Neuronnes par couche
* Avec et Sans Dropout
* Différents optimizers : Adam, Adagrad

Layer (type) Output Shape Param #

=================================================================

conv2d\_31 (Conv2D) (None, 256, 256, 64) 640

max\_pooling2d\_14 (MaxPooling (None, 128, 128, 64) 0

conv2d\_32 (Conv2D) (None, 128, 128, 32) 18464

max\_pooling2d\_15 (MaxPooling (None, 64, 64, 32) 0

conv2d\_33 (Conv2D) (None, 64, 64, 16) 4624

max\_pooling2d\_16 (MaxPooling (None, 32, 32, 16) 0

conv2d\_34 (Conv2D) (None, 32, 32, 8) 1160

max\_pooling2d\_17 (MaxPooling (None, 16, 16, 8) 0

conv2d\_35 (Conv2D) (None, 16, 16, 4) 292

max\_pooling2d\_18 (MaxPooling (None, 8, 8, 4) 0

flatten\_4 (Flatten) (None, 256) 0

dense\_12 (Dense) (None, 256) 65792

dropout\_8 (Dropout) (None, 256) 0

leaky\_re\_lu\_8 (LeakyReLU) (None, 256) 0

dense\_13 (Dense) (None, 256) 65792

dropout\_9 (Dropout) (None, 256) 0

leaky\_re\_lu\_9 (LeakyReLU) (None, 256) 0

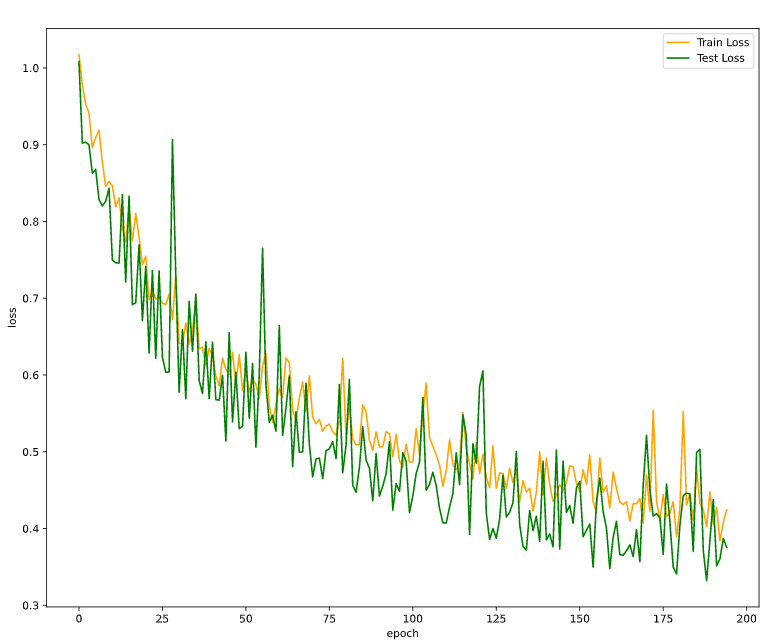
dense\_14 (Dense) (None, 4) 1028

=================================================================

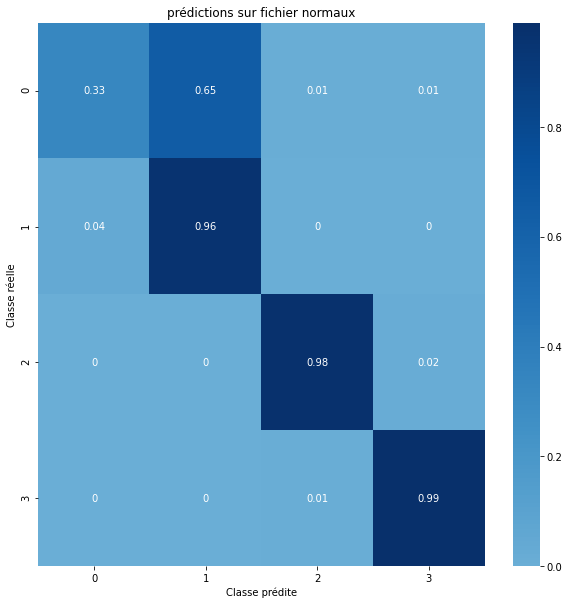
Total params: 157,792

Trainable params: 132,612

Non-trainable params: 25,180



La subtilité de classifier les le type de son à partir de la qualité des prédictions “bonnes” vs “mauvaises” consiste à déterminer dans quel cas on considère que la prédiction est “bonne” et dans quel cas celle-ci est “mauvaise”. Dans un premier temps nous avons essayé d’effectuer cette classification de manière très simple: si l’ID prédite n’est pas la bonne alors la prédiction est mauvaise et le fichier anormal.



74 erreurs sur la classe normal soit 18.50 %

528 erreurs sur la classe anomaly soit 59.33 %

détails fichiers normaux

precision recall f1-score support

0 0.89 0.33 0.48 100

1 0.60 0.96 0.74 100

2 0.98 0.98 0.98 100

3 0.97 0.99 0.98 100

accuracy 0.81 400

macro avg 0.86 0.81 0.79 400

weighted avg 0.86 0.81 0.79 400

détails fichiers anomaly

precision recall f1-score support

0 0.06 0.01 0.01 356

1 0.68 0.37 0.48 267

2 0.70 0.97 0.81 178

3 0.19 0.98 0.32 89

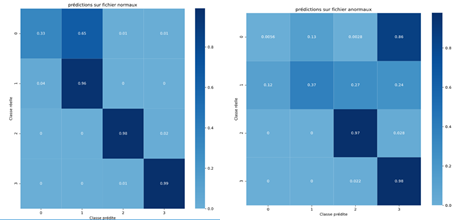
accuracy 0.41 890

macro avg 0.41 0.58 0.41 890

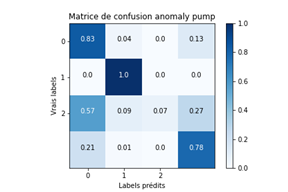
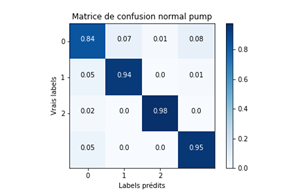
weighted avg 0.39 0.41 0.34 890

Selon les machines :

fan: slider plutot non?



pump :

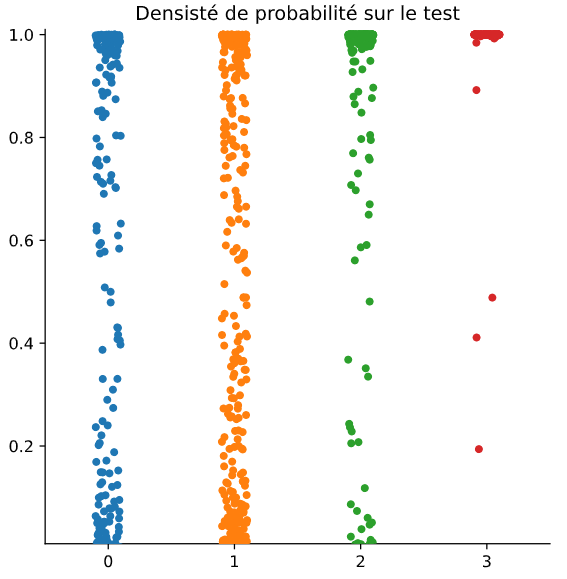
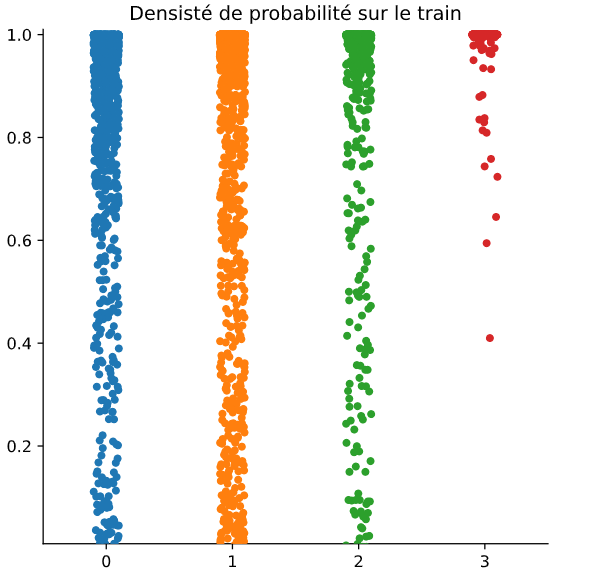


En générant une matrice de confusion entre vrai labels et labels prédits sur le jeu de données test de la machine fan comportant des anomalies nous avons remarqué les problèmes suivants:

1. Le modèle devant prédire 4 classes, la probabilité d’une classe devient majoritaire au dessus de 25%. L’impact d’une chute de probabilité n’aura pas forcément la même incidence sur le classement si la certitude associé à l’ID de base est forte ou faible. Le modèle ne prend donc pas en compte cette chute de probabilité
2. Le paradoxe de l'entraînement. Plus le modèle s'entraîne, plus il renforce ces certitudes sur la détection d’ID normaux, faisant de ce fait monter la densité de probabilité associée. Toutefois, cette certitude devenant forte, l’impact du son anormal n’est plus suffisant pour générer un défaut de classement.

Dans le but de pouvoir travailler sur une chute de précision plutôt que sur une mauvaise prédiction nous avons visualisé les densité de probabilité en sortie du modèle pour chaque véritable ID en entrée.

Les figures représentes donc la valeur de la sortie (softmax) pour le label associé en entrée. Nous pouvons voir sur le cas suivant que le label ‘3’ est très bien catégorisé par le modèle, car la probabilité d’être cet ID est globalement supérieure à 0.9 lorsqu’il est présenté au modèle. A l’inverse l’ID ‘1’ est très étalé, nous pouvons donc voir que le modèle à plus de difficultés à l’identifer.



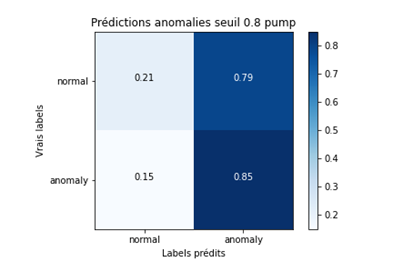
Nous avons donc dans un premier temps fixé un premier seuil identique à chaque ID à la lecture des graphiques.

avec le seuil de 0.25 :

proportion de fichiers normaux mal classés: 7.25 %

proportion de fichiers anormaux mal classés: 30.48 %

En modifiant ce seuil à 0.8 :



5/ Modèle de classification des ID de type RNN

Afin d’améliorer la catégorisation et la fluidité de l'entraînement nous avons essayé de nous passer d’auto-encoder. Pour cela nous nous sommes tournés vers des cellules qui prennent en compte une dimension supplémentaire: le temps.

Nous avons donc utilisé des RNN en utilisant directement les données du train afin de garder la temporalité.

Nous avons ensuite fait varier plusieurs paramètres :

* 2 à 3 couches RNN cachées avant la couche de sortie
* Ajouter 1 à 2 couches de Dense entre les RNN et la sortie
* 64 à 1024 Neuronnes par couche
* Avec et Sans batch\_normalization
* Avec et sans dropout / avec et sans recurent\_dropout
* Différents optimizers : Adam, Adagrad

Layer (type) Output Shape Param #

=================================================================

lstm\_38 (LSTM) (None, 256, 64) 82176

batch\_normalization\_33 (Batc (None, 256, 64) 256

leaky\_re\_lu\_48 (LeakyReLU) (None, 256, 64) 0

lstm\_39 (LSTM) (None, 64) 33024

batch\_normalization\_34 (Batc (None, 64) 256

leaky\_re\_lu\_49 (LeakyReLU) (None, 64) 0

dense\_26 (Dense) (None, 64) 4160

dropout\_13 (Dropout) (None, 64) 0

leaky\_re\_lu\_50 (LeakyReLU) (None, 64) 0

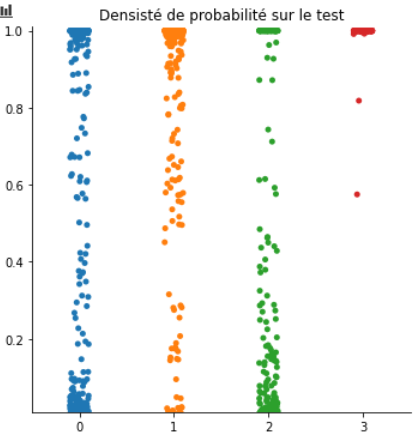
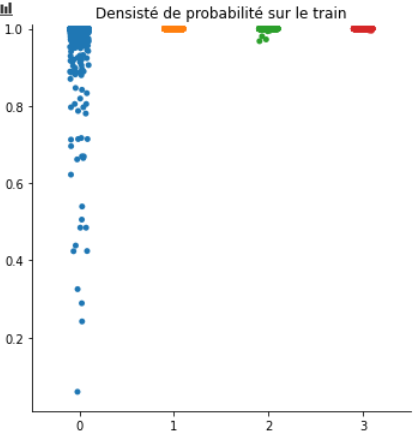
dense\_27 (Dense) (None, 4) 260

=================================================================

Total params: 120,132

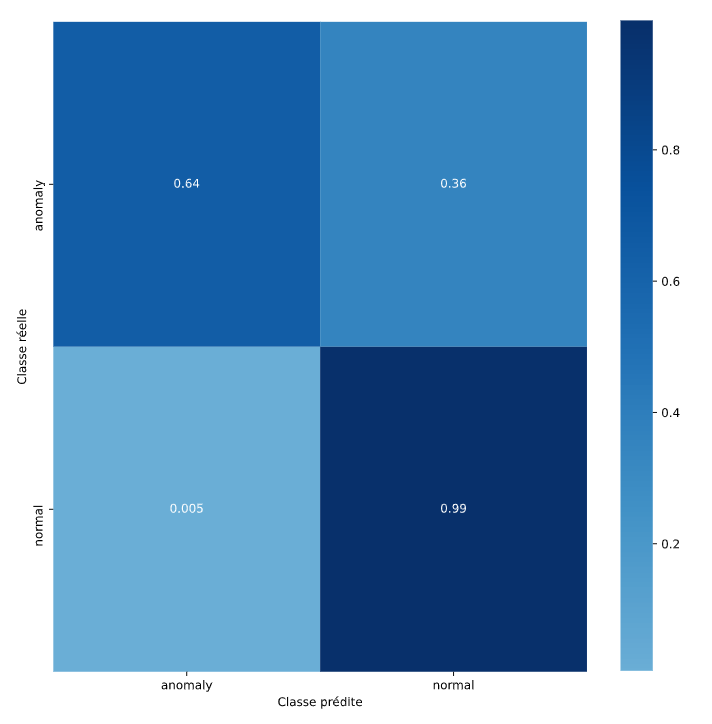
Trainable params: 119,876

Non-trainable params: 256



Nous pouvons donc définir des seuils différenciers par ID listés ci-dessous :

seuil\_ID=np.array([0.4,0.95,0.95,0.95])



# Difficultés rencontrées lors du projet

Nous avons rencontré plusieurs difficultés, que nous pouvons classer dans différentes catégories :

Humaines

La première des difficultés est le travail d’équipe dans un temps restreint sur un domaine qu’aucun d’entre nous ne maîtrise. Il a fallu arriver à se connaître et à se comprendre avant de pouvoir se lancer dans le projet, ce qui en un mois est déjà un premier challenge en soit ! Heureusement, notre bonne humeur à tous et notre envie de travailler ensemble nous ont vite permis de surmonter cette première étape. Une fois ce cap passé nous avons pu commencer à utiliser nos expériences professionnelles très diverses pour amener des visions complémentaires aux problèmes.

Organisationnelles

Au-delà du rythme de la formation à tenir en parallèle du projet, nous avions tous des contraintes liées à la recherche d’emploi ou à un emploi à réaliser en parallèle. Ajoutez à cela les contraintes familiales de chacun et la nécessité de travailler à distance, et nous arrivons à la seconde difficulté. De ce fait, chacun avançait à un rythme différent sur les modules d’une semaine sur l’autre. Les modules étant tantôt très pertinents vis à vis du projet tantôt très éloigné, il était au début difficile d’avancer tous ensemble, nous avons donc mis en place des échanges réguliers via slack et zoom afin d’avoir tous le même niveau d’information sur le projet.

Nous avons également dû apprendre à utiliser des outils de code très différents (Jupyter, Collab, GitHub,VSC) tout en essayant de comprendre comment aller articuler notre projet et donc quels aller être nos futurs besoins. Nous avons réalisé un certain nombre d’aller retours sur des environnements de travail locaux et cloud selon les phases et les difficultés du projet. Cela a mis à rude épreuve l’organisation de chacun et du projet, devant à chaque fois naviguer entre les différents environnements et les données qui y étaient stockées.

Nous avons donc pu apprendre à utiliser plusieurs compilateurs différents, des systèmes locaux et sur le cloud, le tout avec et sans outils de calcul de différence.

Techniques

Au début nos algorithmes étaient simples, et notre connaissance limitée, techniquement nos limites étaient donc le temps de chargement du jeux de données (9Go) qui au delà de la place prise sur le disque, demandait un temps de traitement de 30-50min avant de pouvoir être utilisé. Nous avons réussi avec le temps à contourner le problème de temps de traitement en mettant en place des fichiers npy prétraités, mais pas la problématique de la dimension du jeux de données.

Très vite la taille de ce dernier nous a posé un nouveau problème: l'impossibilité de le stocker dans son intégralité sur le cloud. Nos machines locales étant limitées en puissance de calcul, nous avons donc une double limitation qui nous a obligées à traiter les données machine par machine au lieu de traiter tout la donnée ensemble. De même cette capacité de calcul limitée nous a obligé à bâtir des modèles au plus juste de notre besoin, l’ajout de dimension nous renvoyant immédiatement une erreur de puissance disponible.

Cela nous a permis de découvrir deux axes importants de la DS, des systèmes afin de stocker des données pré-traitées et les charger ainsi que la recherche d’hyper paramètres plutôt que le renforcement de la complexité du réseau.

Expérience

Enfin l’objectif premier du projet étant d'acquérir de l’expérience sur les processus de machine learning et sur les environnements associés il était naturel qu’au début aucun d’entre nous n’avait d’expérience solide dans ce domaine. Toutefois, sans ‘sachant’ dans l’équipe pour nous guider, le choix de la direction n’était pas évident et les aller retours se faisaient au fil des découvertes et des résolutions de problèmes.

Grâce aux erreurs réalisées au fil du projet nous savons un peu plus aujourd’hui comment nous aurions guider le démarrage, le temps et, les ressources que nous aurions accorder à chaque étape.

# Bilan et suite du projet

Une piste d’ouverture pourrait être un dispositif (type Raspberry Pi équipé d’un micro) configuré pour réaliser cette tâche.

# Titre de l’en-tête

|  |
| --- |
| Texte du sous-titre ici Pour commencer immédiatement, appuyez simplement sur le texte d’un espace réservé (tel que celui-ci), puis commencez à taper pour remplacer ce texte par le vôtre.  Vous voulez insérer une image à partir de vos fichiers ou ajouter une forme, une zone de texte ou un tableau ? Procédez comme suit : Sous l’onglet Insertion du ruban, appuyez simplement sur l’option souhaitée. |
|  |
| Pour commencer immédiatement, appuyez simplement sur le texte d’un espace réservé (tel que celui-ci), puis commencez à taper pour remplacer ce texte par le vôtre.  Vous voulez insérer une image à partir de vos fichiers ou ajouter une forme, une zone de texte ou un tableau ? Procédez comme suit : Sous l’onglet Insertion du ruban, appuyez simplement sur l’option souhaitée. |

Iugiiugiugiugdf fd fd fd fd fd fd

dfdfdf