DataScientest • com

Rapport Technique d'évaluation

Détection de son anormal dans les pièces industrielles

Promotion Bootcamp DS - Octobre 2020

Participants :
Abel Traore
Clément Romerowski

Nous ne pourrons commencer la rédaction de ce rapport sans adresser à Thomas Boehler nos remerciements pour ses conseils, son accompagnement tout au long de la réalisation de ce projet. Nous remercions aussi toute l'équipe DataScientest pour la qualité des cours sur la plateforme et leur disponibilité durant toute la formation.

Contexte

Dans le milieu industriel, anticiper les pannes de machines est un enjeu majeur. En effet, les pannes sont source de pertes, et les éviter est un objectif courant, bien qu'elles soient inéluctables. Parvenir à les anticiper afin d'effectuer la maintenance nécessaire le plus efficacement possible (en termes de temps et de coût), de manière préventive plutôt que curative, permet des économies non négligeables.
Toutefois, dans un cas comme dans l'autre, pour être en capacité de prendre les mesures qui s'imposent au plus vite, il faut être en possession des informations concernant l'état des machines. La surveillance humaine possède ses défaillances, et il est possible d'automatiser ce monitoring. Mais quelles données surveiller ? L'image pourrait en être une, mais ni optimale ni bon marché. On pourrait également surveiller les statistiques de production des machines, mais cela ne permettra pas forcément d'anticiper les pannes, et surtout, cette méthode ne peut être généralisée, dans la mesure où elle n'est pertinente que pour les machines directement liées à la production. Une donnée pertinente, généralisable, bon marché et peu coûteuse, est le bruit généré par des machines. En effet, celui-ci porte plus d'informations qu'on ne pourrait l'imaginer : toute défaillance ou faiblesse, qu'elle soit mécanique ou électrique, se traduit par une variation de bruit. On peut ainsi détecter l'usure d'une pièce, sa rupture, ou tout autre changement de condition de fonctionnement par analyse acoustique.
L'acquisition du bruit d'une machine pouvant s'effectuer simplement par l'usage d'un micro, la difficulté réelle est ensuite d'analyser ce bruit pour en tirer les informations qui nous intéressent. C'est ici qu'intervient le *machine learning* : l'idée est d'analyser un ensemble de bruits normaux d'une machine pour en extraire les caractéristiques qui les constituent, afin de savoir les reconnaître. Une fois ceci fait, on peut rechercher les mêmes caractéristiques dans des bruits de la même machine dont on ne connaît pas la nature (normale ou anormale), et en fonction des caractéristiques trouvées ou non, prédire si le bruit résulte d'un fonctionnement normal ou non de la machine.
Les données audio sont particulièrement complexes, dans la mesure où un enregistrement ne contient que l'image des variations de pression acoustique au cours du temps. Ces données sont très peu interprétables en elles-mêmes, et il faut les transformer pour visualiser l'information fréquentielle, plus parlante. Mais comme de faibles variations peuvent rendre compte de phénomènes d'importance non négligeable, la transformation effectuée doit conserver une précision suffisante. Nous avons donc à faire à des données d'entrée possédant de nombreuses *features* (plusieurs milliers), aux corrélations pas toujours évidentes à discerner, ce qui nécessite la mise en œuvre d'algorithmes de *deep learning* ainsi que des ressources conséquentes pour les calculs. Ces applications sont assez récentes pour l'audio, et nous avons pu constater à quel point chaque paramètre a son importance dans l'obtention de résultats pertinents.

Data

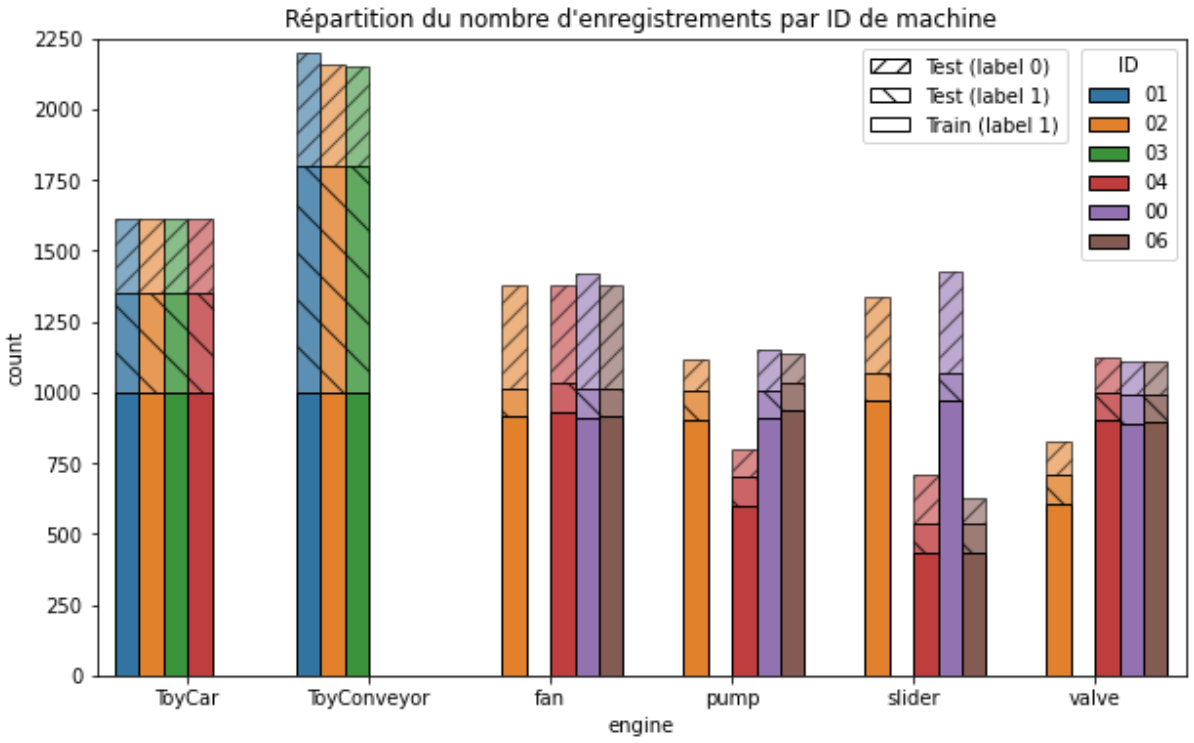
Cadre

L'idée de notre problématique prend sa source dans l'expérience de Clément en tant qu'acousticien de formation. Son souhait de travailler avec des données audio l'a amené à découvrir l'existence d'un challenge annuel de *Detection and Classification of Acoustic Scenes and Events* (DCASE) ; les données utilisées dans le cadre de ce projet proviennent de l'édition 2020 de ce challenge, dont l'une des tâches était de prédire la normalité ou non de différentes machines à partir de leur bruit. Une copie non officielle de ces données est mise à disposition sur kaggle.
Le jeu de données est constitué de 30 986 fichiers audio (au format *.wav, mono, échantillonnés à 16 kHz, d'une durée d'au moins 10 s). Ces données concernent 6 types de machines (ToyCar, ToyConveyor, fan, pump, slider, valve), chacune représentée par 3 ou 4 machines différentes identifiées par un ID. L'ensemble est séparé en deux dossiers :

- train : qui ne contient que des données normales (label 1)
- test : qui mélange des données normales et anormales (label 0).

On labellise les données par 0 (anormales) ou 1 (normales).

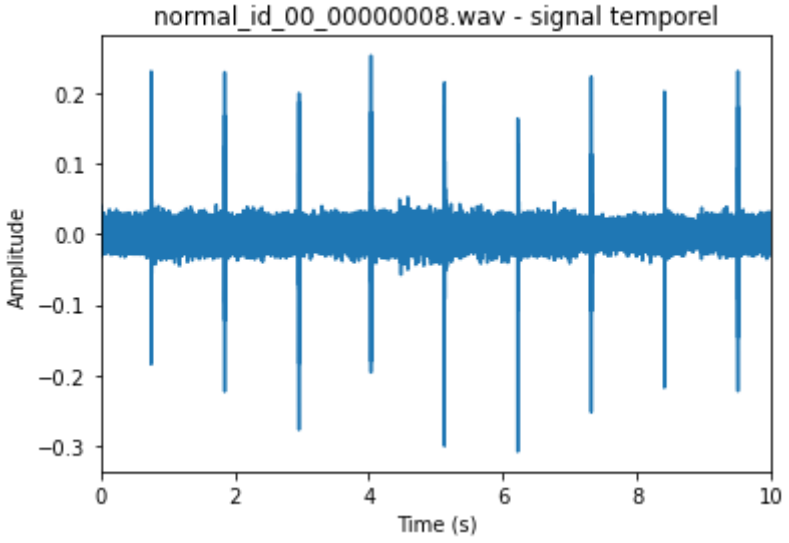
Les données sont réparties ainsi :



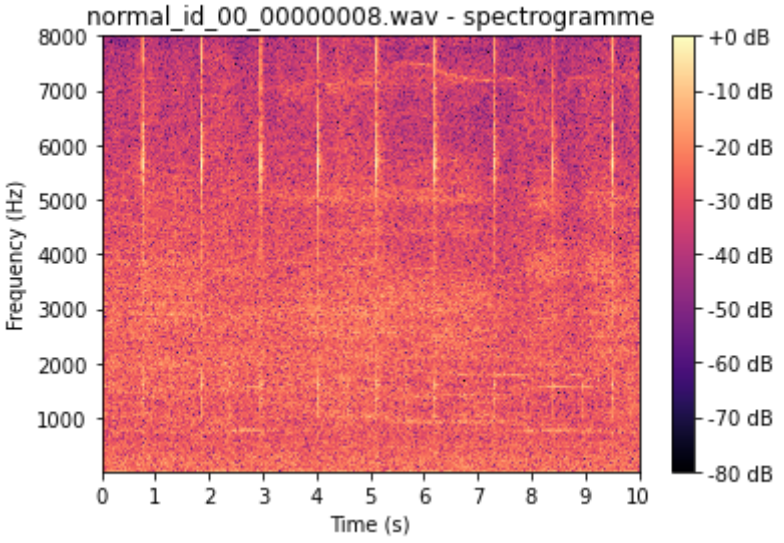
On constate que la répartition est globalement équilibrée, dans la mesure où les ordres de grandeur sont relativement semblables pour chaque machine. On s'étonne toutefois de trouver plus de données anormales que normales dans l'ensemble train, alors que les problématiques de détection d'anomalies considèrent généralement les anomalies comme des exceptions.

Pertinence

Les enregistrements audio dont nous disposons durent tous 10s, à l'exception de ceux du ToyCar qui font 11s. Par souci d'homogénéisation des données pour en faciliter le traitement, nous avons tronqué ces dernières pour avoir un jeu de données uniforme contenant des enregistrements de 10s.
Comme annoncé en introduction, les données audio sous forme de signal temporel ne sont pas exploitables en tant que telles, et doivent être transformées. En effet, l'exemple de la figure ci-après, représentant le signal temporel d'un enregistrement issu du dossier valve, ne nous permet pas de caractériser le bruit autrement que par des impacts réguliers.



La décomposition en spectrogramme fréquentiel est la plus courante ; elle consiste à transformer le signal audio en spectre fréquentiel par FFT, en fenêtrant le signal audio pour conserver l'information temporelle.
Suivant les choix effectués (nfft de 512, overlap de 256), cette transformation permet d'augmenter l'information exploitable sans augmenter significativement le nombre de *features*, qui passe de 160 000 (pour des signaux de 10s échantillonnés à 16kHz) à 513x313, soit 160 569.

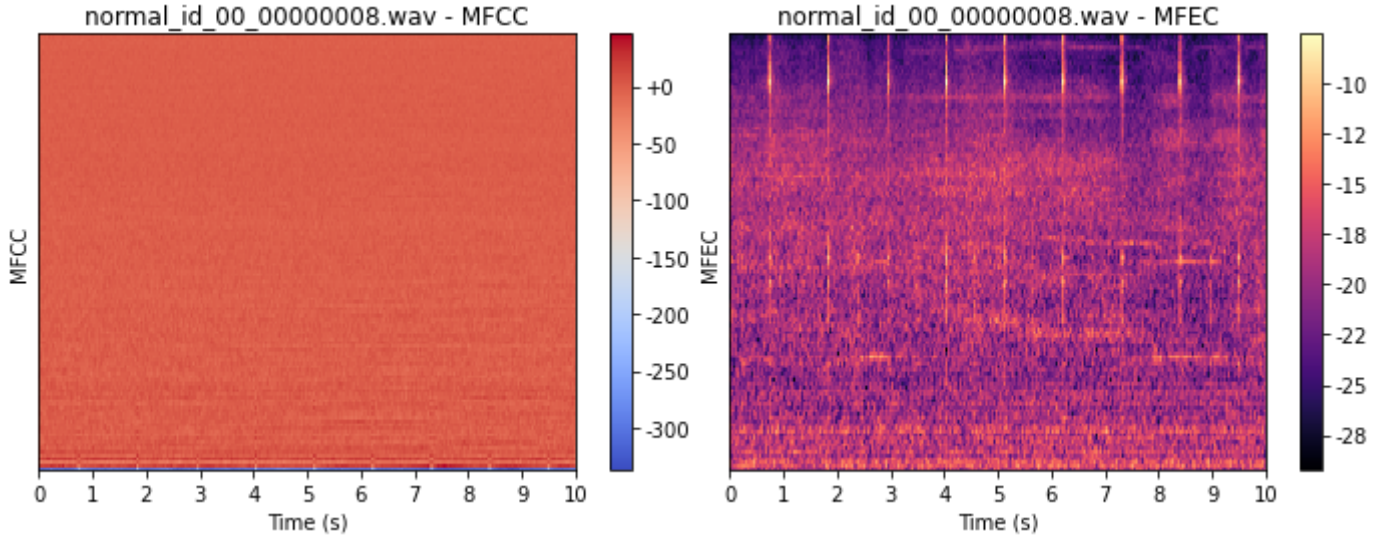


On retrouve les impacts réguliers sur cette représentation, avec d'autres informations : le bruit de fond se situe majoritairement sous 4000Hz, et on observe quelques tonalités qui apparaissent de manière irrégulière.

Nous avons ensuite pensé à utiliser l'échelle de Mel. Cette échelle, conçue comme une échelle psychoacoustique de hauteur des sons (par des personnes n'étant visiblement pas musiciens, puisque sa définition vient en contradiction avec la théorie de la musique), est liée aux hertz par dépendance logarithmique. Elle rend ainsi mieux compte de ce qui se passe dans les graves par rapport aux aigus, et son étendue est plus réduite. On peut alors réduire le nombre de coefficients obtenus par filtrage sans perte d'information importante dans notre cas.

Ayant connaissance de l'usage de la transformation d'un signal en Mel-frequency cepstral coefficients (MFCC), nous avons d'abord opté pour ce choix, avant de découvrir dans une publication les MFEC, (Mel-frequency energy coefficients) qui sont construits de manière semblable à l'exception de la transformée discrète en cosinus des MFCC qui supprime la corrélation entre les coefficients. Cette dernière peut néanmoins avoir son importance suivant le modèle de *deep learning*. C'est pourquoi nous nous sommes finalement orientés vers l'utilisation des MFEC pour représenter nos données d'entrée.

On peut d'ailleurs constater sur la figure ci-après que la représentation par MFCC semble contenir bien moins d'informations (on ne retrouve les impacts que sur les premiers coefficients). En revanche, la représentation par MFEC ne semble pas bien différente du spectrogramme, à ceci près que les 128 coefficients utilisés ici tronquent le signal dans les hautes fréquences : il faudrait le comparer à un spectrogramme dont l'axe fréquentiel s'arrêterait vers 6500 Hz. Il semble toutefois y avoir plus de contraste dans les couleurs, ce qui indique une plus grande disparité dans les valeurs, facilitant probablement l'extraction des caractéristiques du bruit.



On note que cette représentation compte 40 064 valeurs (128x313).

Pour optimiser le fonctionnement de nos modèles, nous avons effectué une étape de *scaling* sur l'ensemble de nos données, à partir des données d'entraînement.

Projet

Classification du problème

Notre problématique est la prédiction sur la normalité de fichiers audio. Il semble donc qu'on peut l'apparenter à un problème de *clustering*, dont la mise en œuvre nous permettra de regrouper les données normales, et nous considérerons alors les *outliers* comme des anomalies. Ainsi, c'est bien une tâche de détection d'anomalie qui est la nôtre.
Toutefois, comme décrit dans la suite, nous avons commencé par traiter notre problématique comme une tâche de classification.

Choix du modèle et optimisation

Notre problématique est spécifiée dans une première mesure comme celle d'une classification. Cette première approche nous amène à tester un réseau de neurones convolutif (*Convolutional Neural Network - CNN*) à l'aide du package **Keras** afin d'observer si cette architecture est plus adaptée à notre problème de classification. Cette modélisation s'effectue par type de machine en n'utilisant que le jeu de données test. Les types de machines pump, slider et Toy Car nous donnent des résultats satisfaisants. En effet, le modèle prédit les deux classes pour ces types de machines. Par exemple, pour le type de machine slider dont le jeu de données comporte 1290 audios, on trouve un score de 79% et une matrice de confusion permettant de détecter 156 vrais audios anormaux et 48 vrais audios normaux. Le f1-score est de 85% pour la classe audios anormaux (label = 0) et de 64% pour la détection des audios normaux (label = 1).
Quant aux 3 autres à savoir fan, valve et ToyConveyor, le modèle ne prédit qu'une seule classe. Donc, ne serait-il pas souhaitable de penser à d'autres idées de classifications afin de pouvoir prédire les 2 classes pour toutes les machines? En effet, même si notre modèle avait permis de prédire les 2 classes pour les 6 types de machines, cette méthode ne pouvait pas nous permettre d'atteindre l'objectif car le jeu de données utilisé pour l'entraînement du modèle ne devrait contenir que des audios normaux. C'est pourquoi, tout en pensant notre problématique comme une tâche de classification, nous allons utiliser d'autres approches permettant de mieux discriminer nos audios en effectuant une modélisation par réseau de neurones.

Nous avons alors imaginé une solution un peu différente : en entraînant un classifieur sur un jeu de données normales représentant 23 classes, nous avons pensé qu'il serait capable de classifier les données test, avec une probabilité plus ou moins élevée, nous permettant par là de discriminer les fichiers anormaux par une probabilité plus faible. En utilisant un réseau de neurones profond convolutif (CNN) inspiré d'une publication [8] dont l'architecture est détaillée en annexe, cette approche s'est révélée fructueuse, dans la mesure où nous avons pu obtenir un ROC-AUC moyen de 82%, et un f1-score moyen de près de 93%.

Cette méthode de diagnostic, bien que permettant d'obtenir un très bon résultat, ne nous a pas semblé très pertinente dans la mesure où le seuil ainsi déterminé est très dépendant des données. Dans l'idée de trouver une méthode plus robuste, l'étape suivante a consisté à

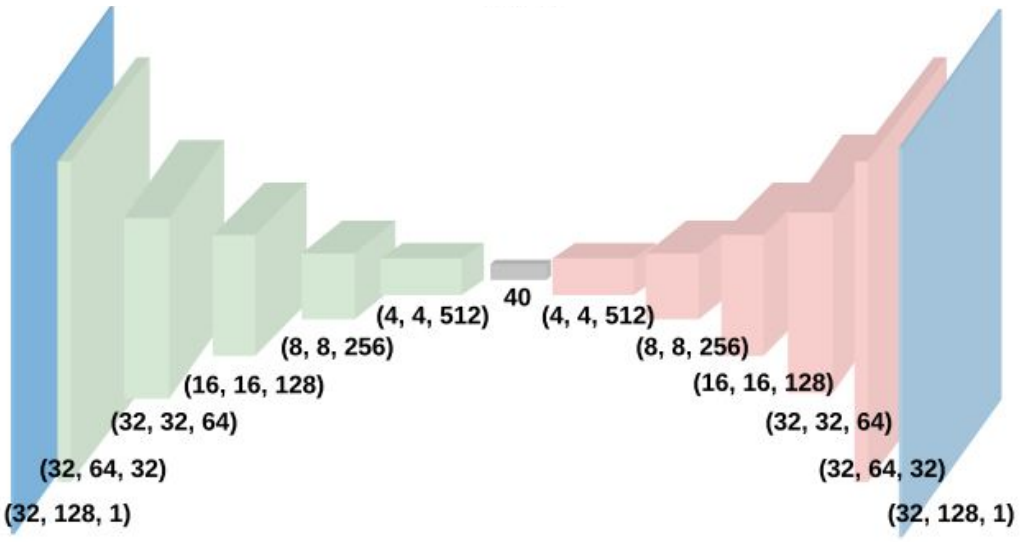
mettre en œuvre une tâche de clustering en partant de l'*embedding* du modèle précédent, un vecteur aux dimensions réduites par rapport aux dimensions des données d'entrée, situé juste avant la couche finale de classification. Cette tentative de clustering par la méthode des k-Means, basée sur [5], a permis d'obtenir des résultats assez bons, présentés ci-après.

engine	ID	f1-scores
ToyCar	1	73,62
	2	80,65
	3	78,86
	4	95,77
moyenne		**82,23**
fan	0	53,65
	2	97,82
	4	66,74
	6	77,22
moyenne		**73,86**
slider	0	94,52
	2	56,68
	4	97,84
	6	88,36
moyenne		**84,35**

engine	ID	f1-scores
ToyConveyor	1	78,67
	2	51,43
	3	57,8
moyenne		**62,63**
pump	0	50,62
	2	73,93
	4	92,5
	6	69,31
moyenne		**71,59**
valve	0	83,11
	2	50,45
	4	83,64
	6	70,45
moyenne		**71,91**
moyenne globale		**74,94**

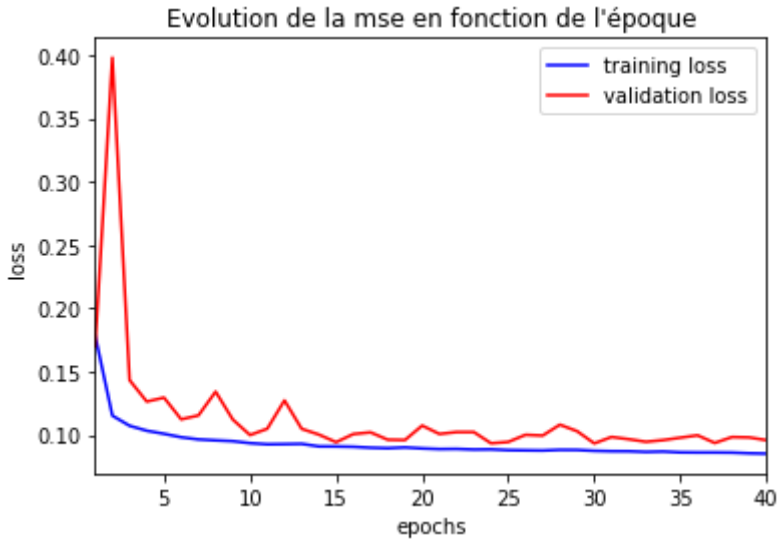
Nous avons ensuite essayé d'autres choses pour améliorer ces résultats : des modèles de type PCA ou OneClassSVM. Ces tentatives n'ont pas été très concluantes.

Après avoir tenté de mettre en œuvre ce que nous avions pu apprendre au cours de la formation, nous nous sommes orientés vers quelque chose de nouveau pour nous, les auto-encodeurs. Initialement utilisés pour réduire les dimensions des données d'entrée, ceux-ci peuvent aussi être utilisés comme débruiteurs, ou comme détecteurs d'anomalies. Cette dernière fonction est celle qui nous intéresse. On entraine auto-encodeur sur des données normales, c'est-à-dire qu'on va réduire les dimensions des données d'entrée, puis effectuer le chemin inverse pour retrouver ces dimensions à partir du vecteur réduit (encodé). On encode donc, puis décode nos données d'entrée, en cherchant à minimiser l'écart entre les données d'entrée et les données de sortie (données reconstruites) - on parle donc d'erreur de reconstruction. La métrique utilisée pour caractériser cette erreur est la *mean squared error*.
Ce faisant, l'auto-encodeur va extraire les caractéristiques principales des données qu'il apprend à reconstruire. Puis, en appliquant le modèle aux données test, les données normales, semblables aux données d'entraînement, devraient produire une faible erreur. En revanche, les données anormales sont censées générer une erreur de reconstruction plus importante, c'est ce qui nous permettra de les discriminer.
Après avoir essayé sans succès de multiples architectures d'auto-encodeurs, denses ou convolutifs, nous avons finalement décidé de mettre en œuvre une architecture détaillée dans une publication qui traite le même problème que nous [9]. Il s'agit d'un auto-encodeur à 5 couches de convolutions qui compresse les données d'entrée en vecteur de dimension 40 avant d'effectuer la transformation inverse, structuré comme suit :

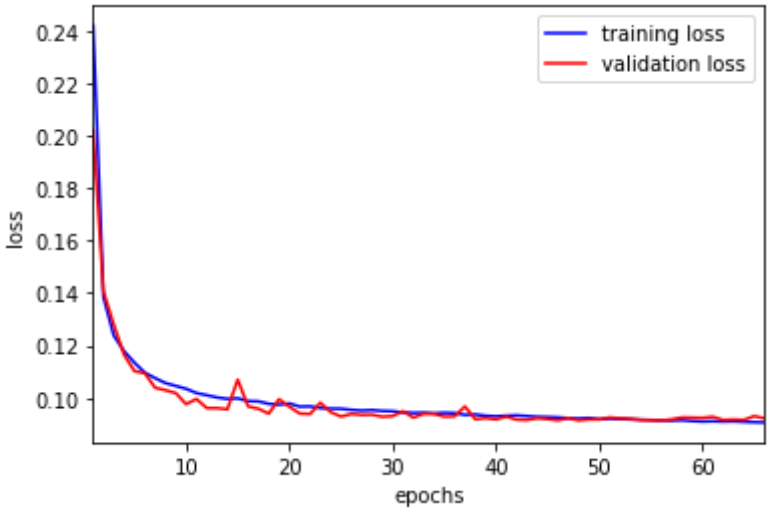


Chaque couche de convolution (2D) subit une normalisation (BatchNormalization), et une fonction d'activation ReLU. Une vision plus détaillée est donnée en annexe.
Comme on peut le remarquer, les tailles d'entrée et sortie de cet auto-encodeur sont bien inférieures à la taille de la représentation par MFEC d'un signal de 10 secondes (128x313). Nous avons découpé le signal en fenêtres de 32 échantillons, qui se recouvrent de 4 échantillons, séparant ainsi le signal original en 11 tranches. C'est sur ces tranches que le modèle a travaillé, et nous avons reconstruit le signal en sortie du modèle pour lui rendre ses dimensions originales.
Nous avons utilisé un optimiseur Adam, avec un taux d'apprentissage de 0.001. Un *callback* a été utilisé pour modifier cette valeur au cours des époques, sans amélioration notable.
Un autre *callback* a été utilisé pour permettre de réduire le temps d'apprentissage après une stabilisation de la *validation loss* sur 10 époques.
Le modèle a été entraîné pour chaque machine (soit 23 fois), sur un maximum de 100 époques, avec des batches de taille 64.

L'observation de la convergence des erreurs d'entraînement et de validation nous a conduits à insérer des couches de Dropout après chaque convolution, pour réduire le surapprentissage que l'on pouvait noter. Cette amélioration sur la convergence s'est bien entendu assortie d'une amélioration sensible des résultats.



sans dropout



avec dropout

Pour chaque modèle présenté ci-dessus, nous avons étudié les erreurs de diagnostic, principalement en comparant à l'écoute les fichiers normaux et les fichiers mal classés. Il s'avère que la différence ne nous a pas souvent semblé audible, ce qui ne nous a pas aidés à améliorer les modèles.

Les deux derniers modèles nous ont permis d'obtenir des résultats corrects, qui sont présentés en annexe sous la forme de scores ROC-AUC. Nous les comparons à la *baseline*, les scores initiaux communiqués lors du challenge. Globalement, nos résultats sont meilleurs que ceux-là, bien qu'il arrive que nos modèles aient moins bien performé. Si l'on fait une moyenne des scores, le classifieur CNN obtient les meilleurs résultats. On remarque toutefois que suivant le type de machine, en particulier le ToyCar et le ToyConveyor, l'auto-encodeur est plus performant.

Difficultés rencontrées lors du projet

La première difficulté à laquelle nous avons été confrontés a été la transformation des données audio brutes en données exploitables contenant les informations fréquentielles. Bien que Clément soit coutumier de ces pratiques, ça n'a pas été naturel pour Abel.

La seconde difficulté identifiée a été la prise en main des auto-encodeurs. Ces outils, bien que puissants et plutôt couramment utilisés de par leurs fonctions diverses, ne sont pas encore enseignés au cours de la formation, et nous pensons qu'ils gagneraient à l'être. Toutefois, nous pouvons aisément nous les approprier à partir des notions transmises dans la deuxième moitié de la formation, mais c'est plus difficile au début du projet, alors que nos compétences sont encore bien faibles.
Nous avons pu travailler sur un module d'introduction à Keras en début de projet pour prendre en main ces notions, mais nous n'avions pas encore le recul nécessaire pour appréhender pleinement l'usage et la puissance de ces outils.
Le libre accès à divers tutoriels et ressources [1], [2], [3], [4] a été d'un grand secours pour nous.

Enfin, une difficulté plus matérielle cette fois, concerne la machine de Clément, assez limitée pour les tâches que nous devions effectuer. Clément a donc travaillé principalement sur Google Colaboratory, qui est une ressource pratique mais limitée. L'utilisation des GPU n'étant pas systématiquement permise, de longues poses sont parfois imposées avant de pouvoir continuer à travailler. Il regrette de fait de ne pas avoir pu effectuer tous les tests qu'il aurait voulu pour participer à l'optimisation du modèle.
C'est d'ailleurs pour cette raison que les codes sont transmis sous forme de notebook uniquement. Cela présente l'avantage de pouvoir les expliquer et les introduire facilement. Ainsi, le détail de notre travail se trouve dans le notebook associé à ce rapport, également disponible ici.

Bilan

Au cours de nos investigations pour atteindre les objectifs du projet, nous avons été confrontés aux limites des cours que nous avons eu lors de la formation, bien qu'ils soient complets. Cela a été l'occasion d'explorer d'autres techniques peut-être plus adaptées à la détection d'anomalies. Nous avons ainsi pu découvrir les auto-encodeurs. La pratique de cette méthode nous a permis d'avoir des connaissances nouvelles qui pourront être encore approfondies. Au-delà de cette nouveauté, ce projet s'est montré instructif à bien des égards.

D'une part, nos données, du fait de leur format et ce qu'elles représentent, ne nous ont pas permis d'effectuer une fine analyse statistique, mais leur mise en forme pour exploitation a constitué un premier défi pour nous.
D'autre part, la problématique de détection d'anomalies nous a permis de mettre en œuvre plusieurs types de modèles de deep learning : classification, clustering, auto-encodeurs.

Pour les modèles enseignés au cours de la formation, c'était une bonne occasion de les mettre en application et de jouer avec les paramètres pour étudier leur incidence. Concernant les auto-encodeurs, nous avons été amenés à les découvrir par nous-mêmes, et la mise en application directe a été formatrice. Ainsi, nous avons pu développer une meilleure compréhension de différents paramètres des couches (strides sur les couches convolutives par exemple), de l'impact du choix de l'activation, de l'optimiseur, de la fonction de coût, ..., en évaluant leur impact direct sur la convergence ou les résultats du modèle. Ces éléments constituent un début d'expérience qui pourra nous aider à nous orienter dans le métier.
Par ailleurs, l'objectif de ce projet était d'utiliser les connaissances d'intelligence artificielle afin de classifier une machine sous son état normal ou anormal à partir du bruit émis. Dans un premier temps à travers les modélisations par réseaux de neurones convolutifs, nous avons pu classifier les 23 machines après plusieurs tentatives avec différentes méthodes. Bien que nous ayons abouti à une classification par machine, notre problème s'est vite présenté comme celui de détection d'anomalie, donc il s'agissait de distinguer les audios anormaux des autres. Ce nouvel objectif nous imposait d'aller trouver des méthodes en dehors de nos notes de cours. Cela nous a amenés à une modélisation par utilisation des auto-encodeurs, en passant par un clustering sur l'embedding du modèle CNN utilisé pour la classification. Ainsi, les résultats que nous obtenons avec nos deux modèles sont corrects, globalement meilleurs que ceux affichés par la baseline du challenge initial. Bien que l'on puisse considérer le classifieur CNN comme globalement plus performant que l'auto-encodeur convolutif, l'étude des résultats montre que suivant le type de machine, c'est l'auto-encodeur qui est plus pertinent. En tous cas, nous avons une solution qui permet de déterminer si une machine est anormale avec une précision moyenne de 75% (l'étendue des précisions allant de 50% (notre modèle est alors inutile), à 98%).
Donc, au regard des objectifs de classification et de détection d'anomalies sur les différents audios, nous pouvons dire que les objectifs du projet ont été atteints. Cependant, ne serait-il pas possible d'aller plus loin afin de détecter en temps réel automatiquement les anomalies dans chaque son, et avoir une sorte de connecteur installé sur chaque type de machine permettant d'avoir une sorte d'alerte à chaque fois qu'il y a une défaillance ? Et voir si cela permettra d'anticiper et d'activer la maintenance prédictive de façon efficace ?

Les solutions de ce projet doivent résoudre les problèmes liés à la maintenance prédictive dans le milieu industriel. Cependant, le sujet s'approchant directement des problèmes de gestion de risques, il semble que notre modèle pourrait être adapté à certains problèmes rencontrés dans les secteurs banque et assurance (dans la détection des fraudes par exemple).
Cela dit, ce projet étant assez complexe et innovant, pouvant intéresser plusieurs secteurs d'activités, il peut être pertinent de chercher des financements auprès des entreprises afin de le traiter sous forme de thèse CIFRE. Cela permettrait d'avoir plus de temps et de faire beaucoup d'investigations sur la méthode de l'auto-encodeur ou d'autres méthodes afin d'améliorer les performances du modèle. Rappelons néanmoins qu'alors que nos résultats en termes de score ROC-AUC moyen se situent entre 78 et 79% pour nos deux modèles, les meilleurs résultats au challenge dépassent les 90%. Nous avons donc encore du chemin à parcourir !

Bibliographie

Tutoriels pour la prise en main des auto-encodeurs :

[1] **François Chollet**, Building Autoencoders in Keras, 2016, https://blog.keras.io/building-autoencoders-in-keras.html

[2] **Adrian Rosebrock**, Anomaly detection with Keras, TensorFlow, and Deep Learning, mars 2020, https://www.pyimagesearch.com/2020/03/02/anomaly-detection-with-keras-tensorflow-and-deep-learning/

[3] Introduction aux auto-encodeurs, https://www.tensorflow.org/tutorials/generative/autoencoder

[4] Autoencoder for sound data in Keras, https://stackoverflow.com/questions/46816206/autoencoder-for-sound-data-in-keras

[5] Audio feature extraction and clustering https://www.kaggle.com/humblediscipulus/audio-feature-extraction-and-clustering

Publications en rapport avec notre problématique :

[6] **Taha Berkay Duman, Barış Bayram, and Gökhan İnce**, Acoustic Anomaly Detection Using Convolutional Autoencoders in Industrial Processes, https://www.researchgate.net/publication/332795575_Acoustic_Anomaly_Detection_Using_Convolutional_Autoencoders_in_Industrial_Processes

[7] **Yuma Koizumi, Yohei Kawaguchi, Keisuke Imoto, Toshiki Nakamura, Yuki Nikaido, Ryo Tanabe, Harsh Purohit, Kaori Suefusa, Takashi Endo, Masahiro Yasuda, Noboru Harada**, Description and discussion on DCASE2020 challenge Task2: Unsupervised anomalous sound detection for machine condition monitoring, 2020, https://arxiv.org/pdf/2006.05822v2.pdf

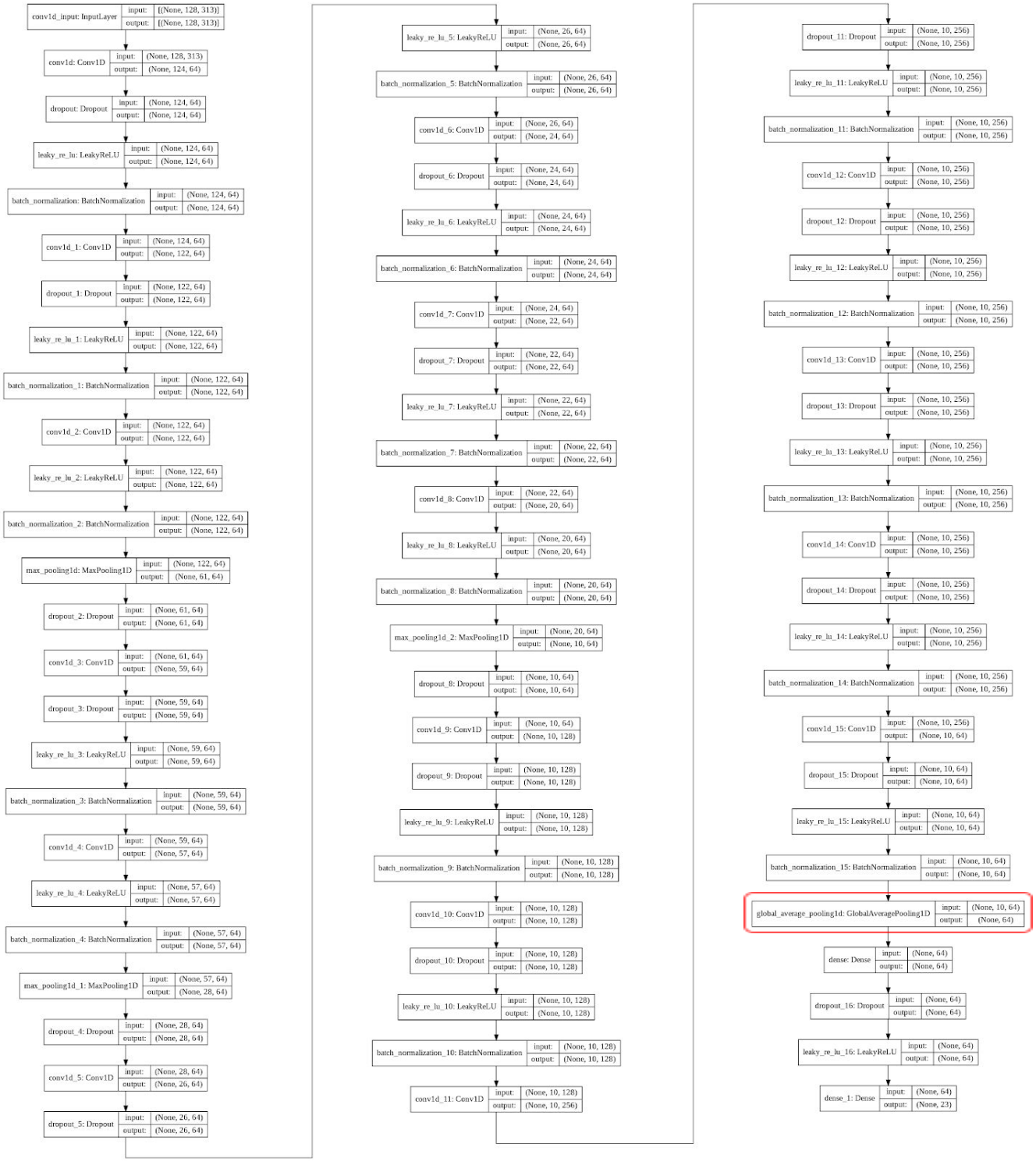
[8] **Paul Primus,** Reframing unsupervised machine condition monitoring as a supervised classification task with outlier-exposed classifiers, Technical report, 2020, http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Primus_36_t2.pdf

[9] **Alexandrine Ribeiro, Luis Miguel Matos, Pedro Jose Pereira, Eduardo C. Nunes, Andre L. Ferreira, Paulo Cortez, Andre Pilastri**, Deep dense and convolutional autoencoders for unsupervised anomaly detection in machine condition sounds, Technical report, 2020, https://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Pilastri_86_t2.pdf

Annexes

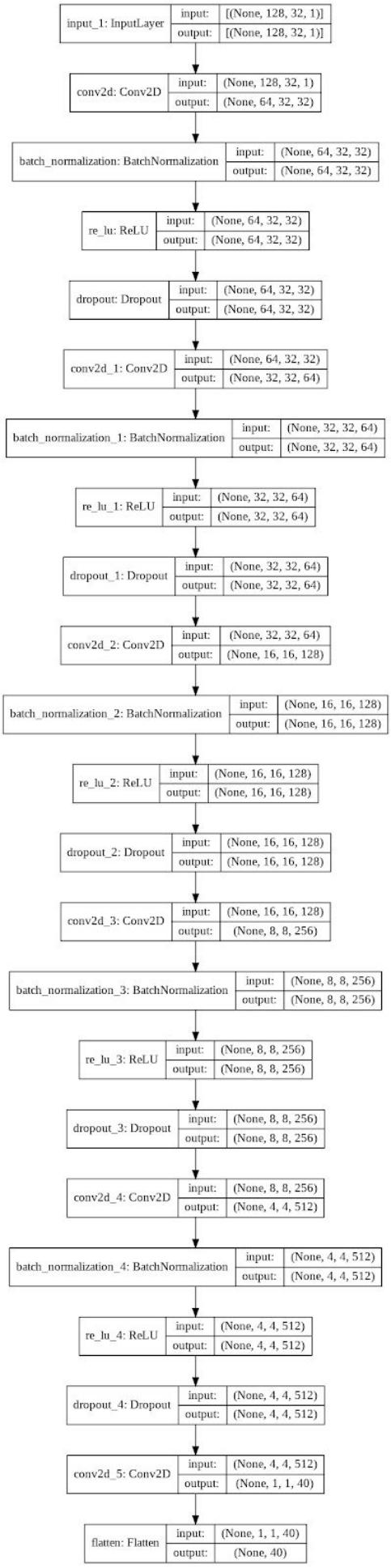
Architecture du classifieur CNN

(La couche de GlobalAveragePooling1D encadrée en rouge est l'embedding utilisé pour le clustering.)

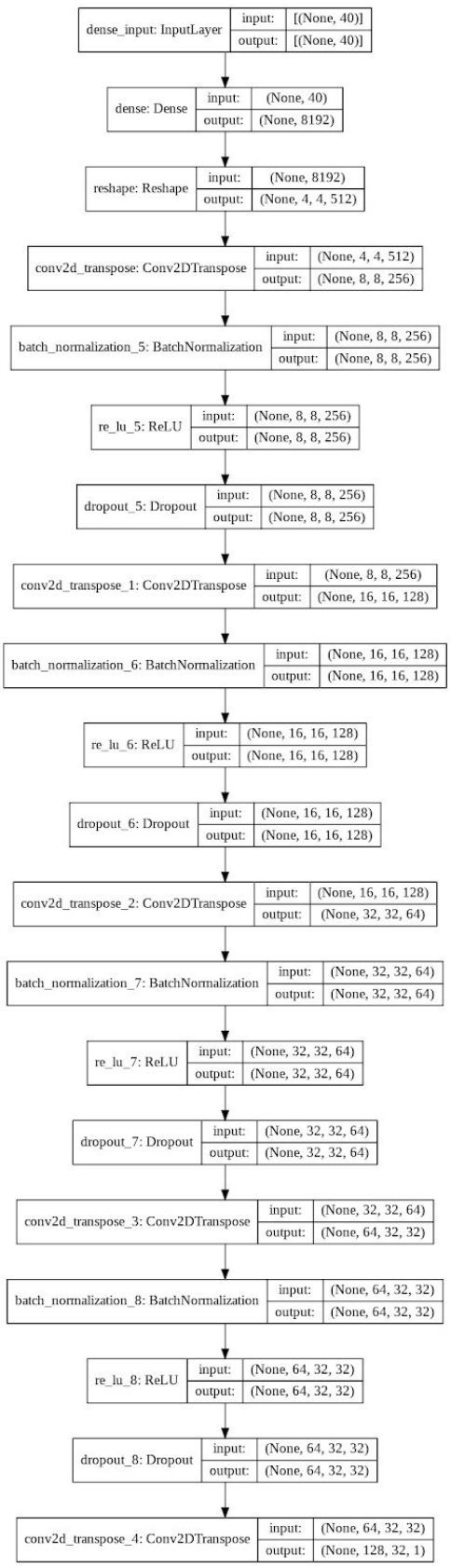


Architecture de l’auto-encodeur convolutif

encodeur



décodeur



Scores ROC-AUC obtenus

(Les scores les meilleurs sont mis en gras.)

engine	ID	scores AE	scores CNN	baseline
ToyCar	1	**82,13**	57,79	81,36
	2	85,76	81,85	**85,97**
	3	64,57	**84,92**	63,30
	4	**86,33**	67,72	84,45
moyenne		**79,70**	**73,07**	**78,77**
ToyConveyor	1	77,34	**77,82**	78,07
	2	**65,28**	58,00	64,16
	3	**77,31**	61,17	75,35
moyenne		**73,31**	**65,66**	**72,53**
fan	0	**55,31**	53,57	54,41
	2	81,49	**99,44**	73,40
	4	63,37	**81,96**	61,61
	6	**91,80**	91,64	73,92
moyenne		**72,99**	**81,65**	**65,84**
pump	0	67,20	**70,18**	67,15
	2	58,21	**78,40**	61,53
	4	**97,64**	94,70	88,33
	6	**77,39**	72,17	74,55
moyenne		**75,11**	**78,86**	**72,89**
slider	0	**98,93**	92,24	96,19
	2	**82,64**	79,83	78,97
	4	94,40	**98,83**	94,30
	6	75,22	**95,25**	69,59
moyenne		**87,80**	**91,54**	**84,76**
valve	0	81,32	**90,42**	68,76
	2	**76,13**	75,68	68,18
	4	**84,03**	74,01	74,30
	6	69,58	**84,80**	53,90
moyenne		**77,77**	**81,23**	**66,29**
moyenne globale		**77,97**	**79,23**	**73,55**