



# Projet Personnel Professionnel

## Maintenance Proactive Dédiée à l'industrie 4.0

Filière: informatique industrielle et automatique

Préparé par :

**Imed Eddine Ben Slimene**  
**Khalil Kaaniche**  
**Mariam Béji**  
**Mohamed Taher Bazzazi**

Soutenue le 06/06/2022, devant le jury:

Prof. Mohamed Ali Hamdi

Examineur

Prof. Mustapha Hamdi

Encadrant

**Année universitaire : 2021/2022**

## Sommaire

Introduction .....	4
Chapitre 1 : Industrie 4.0 .....	5
1- Définition : .....	5
2- Concepts clés de l'industrie 4.0 : .....	5
a- L'usine virtuelle : .....	5
b- L'automatisation des flux : .....	5
c- Les Smart Machine : .....	5
d- Les systèmes de maintenance prédictive : .....	5
e- Cyber système de prédiction : .....	6
3- L'automatisation dans l'industrie 4.0 .....	6
4- La maintenance prédictive dans l'industrie 4.0 .....	6
Chapitre 2: Deep Learning .....	8
1- Les réseaux de neurones .....	8
2- Réseaux neuronaux convolutifs .....	9
3- Couche convolutive .....	10
4- Couche de regroupement ( Pooling Layer ) .....	10
5- Couche de sur-échantillonnage ( Up-Sampling Layer ) .....	11
6- Sauter la connexion .....	11
7- Loss function .....	11
Chapitre 3 : Extraction des données .....	12
1- A partir du microphone .....	12
2- A partir de l'accéléromètre et du gyroscope .....	14
Chapitre 4 : Etude pratique et implémentation .....	15
1- Approche à base du son .....	15
a- Présentation du dataset .....	15
b- Préprocessing .....	16
c- Choix du modèle .....	17
d- Implémentation sur STM32 .....	17
e- Résultat .....	18
f- Amélioration du modèle : .....	19
g- Résultat .....	20
2- Approche à base de la vibration .....	21
a- Présentation du Dataset .....	21
b- Préprocessing .....	21
c- Choix du modèle .....	22
d- Implémentation sur STM32 .....	22
e- Résultat .....	23
Conclusion .....	24
Références : .....	25

## Liste de figures

FIGURE 1 EVOLUTION DE L'INDUSTRIE 4.0 .....	5
FIGURE 2 ELEMENTS DE L'INDUSTRIE 4.0 .....	6
FIGURE 3 DIFFEERENCE ENTRE AI , MACHINE LEARNING ET DEEP LEARNING .....	8
FIGURE 4 RESEAUX DE NEURONNES .....	10
FIGURE 5 SIGNAL D'UN MICROPHONE PDM .....	12
FIGURE 6 L'ECHANTILLONNAGE DU FLUX PDM EN ECHANTILLONS PCM.....	13
FIGURE 7 CALCULE DU LOG-MEL SPECTROGRAMME .....	13
FIGURE 8 LA FORME ORIGINALE DES ONDES SONORES .....	16
FIGURE 9 SPECTRE OBTENUE PAR TRANSFORMATION FREQUENTIELLE.....	16
FIGURE 10 LE MEL-SPECTROGRAMME ORIGINAL.....	17
FIGURE 11 LE SPECTROGRAMME NORMALISE .....	17
FIGURE 12 LE MODELE DE CLASSIFICATION POUR LA PREMIERE APPROCHE.....	17
FIGURE 13 LOGIGRAMME D'IMPLEMENTATION SUR STM32 POUR LA PREMIERE APPROCHE .....	18
FIGURE 14 LE RESULTAT OBTENU .....	18
FIGURE 15 LA DIVISION DU DATASET .....	19
FIGURE 16 LA RELATION ENTRE LE NOMBRE D'ITERATIONS ET LA PRECISION DU MODELE .....	19
FIGURE 17 VISUALISATION DE L'EVOLUTION DE L'ENTRAINEMENT.....	20
FIGURE 18 RESULTAT OBTENU APRES AMELIORATION.....	20
FIGURE 19 RESULTAT DU TEST.....	20
FIGURE 20 LE MODELE DE CLASSIFICATION POUR LA DEUXIEME APPROCHE .....	22
FIGURE 21 LOGIGRAMME D'IMPLEMENTATION SUR STM32 POUR LA DEUXIEME APPROCHE .....	22
FIGURE 22 LE RESULTAT OBTENU .....	23

## Introduction

Ce document comportera un aperçu assez complet du projet de fin d'année pour notre quatrième année en informatique industrielle et automatique au sein de l'Institut des sciences appliquées et de Technologie (INSAT). Le sujet étant « Maintenance proactive dédiée à l'industrie 4.0 », il était question de développer une application à base d'intelligence artificielle dans le cadre de la dernière révolution industrielle en date.

Premièrement, on avait comme tâche de nous familiariser avec l'environnement de développement lié à la carte utilisée « STM32 Discovery kit IoT Node » et de découvrir les différents capteurs présents pour en extraire les données. Deuxièmement, On s'est mis à explorer les grandes lignes de l'intelligence artificielle, plus précisément le Deep Learning. Ce qui nous a tout naturellement conduit à choisir l'application suivante pour notre projet de fin d'année : Une détection d'anomalie des ventilateurs dans un environnement industriel à base de « Deep Learning » . En dernier lieu, il était question de traiter le « Dataset » utilisé, entraîner un modèle de « Deep Learning » et de l'implémenter sur le microcontrôleur.

## Chapitre 1 : Industrie 4.0

### 1- Définition :

L'industrie est par définition le fait de transformer des matières premières ou des matières, ayant subies un ou plusieurs traitements en amont, en un produit. Ceci étant réalisé grâce à des technologies de production de plus en plus avancées répondant au besoin de l'humain. Ces changements au niveau des technologies de production ont été regroupés en 4 grandes « révolutions » allant de l'industrie 1.0 à l'industrie 4.0. En effet, la première révolution industrielle marque le 18ème siècle par l'utilisation des moteurs à vapeur et à charbon. La deuxième quant à elle, qui débute en 1840 à peu près est étroitement liée à l'exploitation de l'électricité. Une troisième révolution verra le jour en 1970 (appelée digital revolution en anglais) permettant l'automatisation complète des chaînes de production via des automates et / ou des robots. Enfin, la quatrième révolution industrielle débute lors du 21 en conséquence du développement des technologies liées à l'IOT (Internet Of Things ou internet des objets en français) et à la robotique.

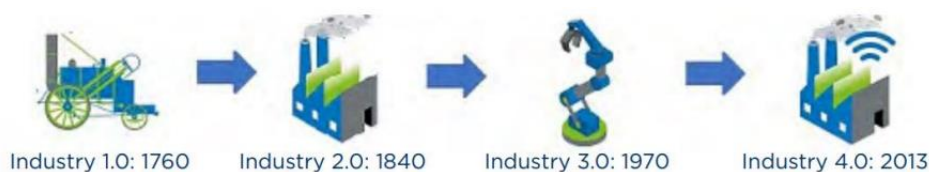


Figure 1 Evolution de l'industrie 4.0

### 2- Concepts clés de l'industrie 4.0 :

Un pas de l'industrie typique à une industrie 4.0 se base sur 5 concepts clés :

#### a- L'usine virtuelle :

Cela se base sur le fait d'utiliser des logiciels de simulation afin de mimer l'industrialisation de nouveaux produits avant le déploiement sur les systèmes dits physiques permettant non seulement d'éviter certains problèmes grâce à la simulation effectuée mais aussi de réduire de manière conséquente le temps de mise au point de la production. De plus cela permet de faciliter la gestion de l'usine sans une présence physique obligatoire

#### b- L'automatisation des flux :

Cela est possible grâce à l'utilisation des robots et des véhicules partiellement ou totalement autonomes. Ceci permet en fin de compte de se passer d'une main d'œuvre humaine étant donné son incapacité à gérer les nouveaux flux et exigences de production (prenons à titre d'exemple l'usine autonome 24/7 d'Okuma).

#### c- Les Smart Machine :

Elles sont des machines indépendantes ne nécessitant pas l'intervention humaine afin de fonctionner correctement. De plus, elles ont la capacité à s'auto-corriger et à s'auto-connecter entre elles. De ce fait, elles prennent en considération leurs différents agissements pour pouvoir fonctionner en conséquent.

#### d- Les systèmes de maintenance prédictive :

Elles offrent la possibilité d'avoir une meilleure planification du temps d'arrêt machines offrant une bien meilleure approche quant à la maintenance. En effet, ces systèmes permettent la prévention vis-à-vis des pannes ou tout autre problème pouvant entraver la production.

#### e- Cyber système de prédiction :

Il permet de faire le lien entre la gestion directe de l'usine, les fournisseurs et les demandes. En effet cela permet d'adapter le plan de production à une variation de demande offrant la possibilité de passer à un modèle d'usine qui stock ses produits à un model différent, celui de la production sur commande.

### 3- L'automatisation dans l'industrie 4.0

L'automatisation dans les temps de l'industrie 4.0 est un processus complexe nécessitant la mise en place de plusieurs couches comme le montre le graphique ci-dessous.

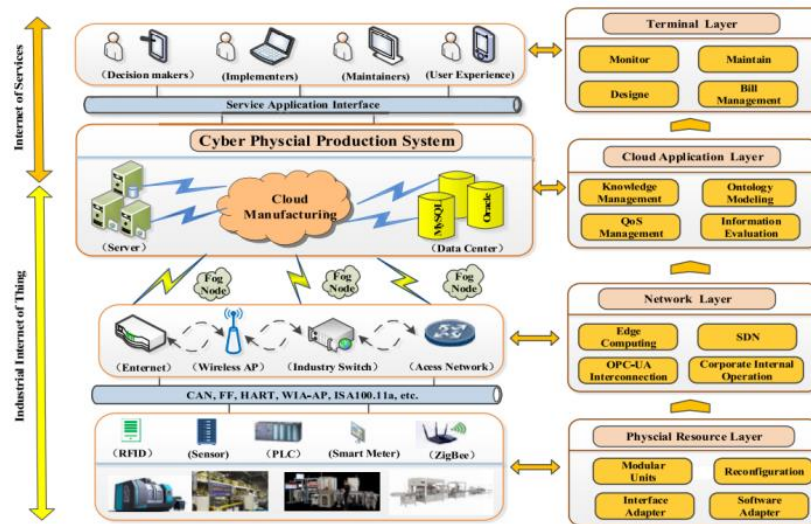


Figure 2 Eléments de l'industrie 4.0

En guise de base, on trouve les capteurs et les actionneurs, représentant respectivement les entrées et les sorties du système. Les flux de données sont gérés par des Automates programmables industriels (appelés PLC en anglais). De plus, on trouve dans cette couche physique d'autres dispositifs comme les points d'accès ZigBee (un protocole de communication bien spécifique caractérisé par une faible consommation énergétique). Ensuite le passage vers la couche réseau se fait via des protocoles de communication (CAN, FF, HART, etc...). On trouve dans cette couche des points d'accès sans fil et un ou plusieurs switches industriels qui permettent de faire parvenir les données à l'étage supérieur dite couche « Cloud Application ». Cette dernière comporte les serveurs et les data centers qui permettent un traitement en masse de l'information récoltée. Un ensemble d'informations ensuite partagé et utilisé par les représentants de la couche « terminal layer » qui est composée des décideurs, des personnes responsables de l'implémentation, du personnel de maintenance, et tout autre intervenant lié à tout le processus. Toute cette structure est mise en place afin de : - Développer la production - Améliorer le contrôle du processus de production - Réduire le besoin d'une main d'œuvre Le tout en analysant les données récoltées ce qui permet une certaine flexibilité et un dynamisme au niveau de la configuration.

### 4- La maintenance prédictive dans l'industrie 4.0

La maintenance prédictive est un type de maintenance qui suit directement l'état de santé, le statut et les performances d'un bien en temps réel. La maintenance prédictive vise à réduire les pannes coûteuses et inattendues et offre au fabricant la possibilité de planifier la maintenance en fonction de son propre calendrier de production.

Grâce à une combinaison de données en temps réel recueillies par l'internet industriel des objets (IIoT), la maintenance prédictive analyse en permanence l'état des équipements pendant les opérations normales afin de réduire la probabilité d'une défaillance inattendue de la machine.

Avec la maintenance prédictive, les organisations peuvent surveiller et tester divers indicateurs tels que la vitesse lente des roulements, la lubrification ou la température. Grâce à la surveillance conditionnelle et à la technologie IIoT, ces outils détectent les anomalies pendant les opérations normales et envoient des alertes en temps réel au propriétaire de la machine qui indiquent une future panne potentielle. Des types plus spécifiques de maintenance prédictive, notamment :

**Analyse des vibrations** : Il s'agit d'un type courant de maintenance prédictive utilisé dans les usines de fabrication de machines rotatives. Elle permet de détecter les déséquilibres, les désalignements ou les pièces desserrées d'un équipement.

**Analyse infrarouge** : En utilisant la température comme indicateur, les problèmes liés à la circulation de l'air, au refroidissement et à la tension du moteur peuvent être identifiés.

**Analyse acoustique sonore** : Les sons peuvent être convertis en un signal auditif ou visuel qui peut être entendu ou vu par un technicien, indiquant des conditions telles que des roulements usés ou insuffisamment lubrifiés dans des machines à faible ou forte rotation.

La maintenance prédictive est une application transformatrice de l'IIoT qui présente des avantages considérables. Ci-dessous, nous explorons cinq avantages :

#### **Diminution des temps d'arrêt**

La maintenance prédictive permet aux techniciens de détecter les problèmes à l'avance et de les résoudre avant que les équipements ne tombent en panne :

Réduire les temps d'arrêt imprévus jusqu'à 30 % programmer plusieurs procédures de service en même temps

Eviter le risque de pannes nuisibles à votre réputation

Réduire les déplacements coûteux de camions dus à des temps d'arrêt imprévus.

#### **Une plus grande productivité des travailleurs**

Il n'est pas nécessaire de perturber la productivité des travailleurs pour un dysfonctionnement ou une panne inattendue. La maintenance prédictive s'adapte aux horaires des travailleurs :

Permet d'accélérer jusqu'à 83 % le délai de résolution des problèmes.

Maximise le temps de fonctionnement et évite les baisses de productivité

Augmente l'utilisation des actifs

#### **Réduction des coûts de service sur le terrain**

En anticipant la maintenance des machines, les services d'entretien peuvent réaliser d'importantes économies et augmenter le retour sur investissement grâce aux éléments suivants :

Réduction des déplacements coûteux des camions de service

Augmentation des taux de réparation dès la première intervention

Rationalisation des coûts de maintenance grâce à la réduction des coûts de main-d'œuvre, d'équipement et d'inventaire.

#### **Improved Product Design**

En exploitant la puissance des données IIoT collectées par les capteurs de votre machine, les concepteurs de produits peuvent utiliser ces informations vitales pour :

Prolonger la durée de vie des actifs

Améliorer la durabilité et la fiabilité des équipements  
Construire des machines plus efficaces à l'avenir

### Amélioration de la sécurité des travailleurs

Une panne ou un dysfonctionnement inattendu peut entraîner des conditions de travail dangereuses pour vos employés. En prévoyant quand un dysfonctionnement peut se produire, vous pouvez vous assurer que :

Les employés ne sont pas à proximité d'une machine lorsqu'elle tombe en panne.

Les techniciens peuvent effectuer des réparations avant que la machine ne devienne dangereuse.

## Chapitre 2: Deep Learning

Tout d'abord nous allons définir l'IA, « Machine Learning » et « Deep Learning ».

Pour résumer, la première consiste à donner à la machine l'aptitude d'imiter le comportement humain (prise de décision ou autre) , la deuxième est un sous-ensemble de l'IA se basant sur les statistiques offrant à la machine la possibilité de s'améliorer au fur et à mesure et la dernière est un sous ensemble du ML se basant sur le principe même des neurones et c'est celui-là qu'on va utiliser dans le projet réalisé.

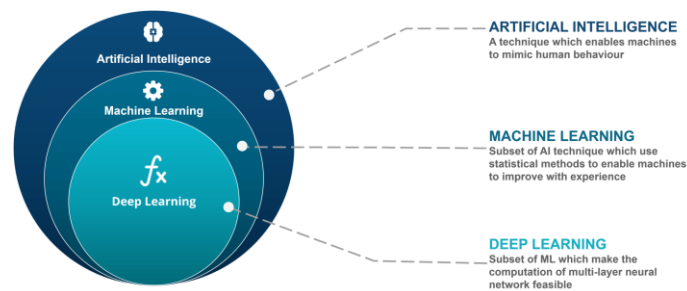


Figure 3 Différence entre AI , Machine learning et Deep learning

Pour parler brièvement du Deep Learning (appelé aussi apprentissage profond) est un ensemble de neurones artificiels constituant des couches. Ces dernières couplées avec une couche d'entrée ainsi qu'une couche de sortie constituent un réseau de neurones.

### 1- Les réseaux de neurones

Les réseaux neuronaux sont composés d'un certain nombre d'unités de calcul connectées, appelées neurones et organisées en couches.

Comme le montre la figure un réseau neuronal typique se compose d'une couche d'entrée où les données entrent dans le réseau, des couches cachées transformant les données au fur et à mesure de leur passage, et une couche de sortie produisant les résultats.

Le réseau est formé pour produire des prédictions utiles en reconnaissant des modèles informatifs dans l'ensemble de données de formation, supervisé par la comparaison des résultats prédits aux étiquettes réelles sous le format d'une fonction objective.

$$\hat{f}(x) = h(w^T x + b)$$

où  $x$  représente le vecteur d'entrée,  $w = (w_1, \dots, w_n)$  est le vecteur de poids et  $b$  est le biais.

$h(-)$  est la fonction d'activation non linéaire. Les fonctions d'activation les plus couramment utilisées sont :



(1) Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

(2) Tanh

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(3) Rectified Linear Unit (ReLU)

$$\sigma(x) = \max(0, x)$$

(4) Leaky ReLU

$$\sigma(x) = \max(0.1x, x)$$

(5) Maxout

$$\sigma(x) = \max(w_1^T x + b_1, w_2^T x + b_2)$$

(6) Exponential Linear Unit (ELU)

$$f(x) = \begin{cases} x & , \quad x \geq 0 \\ \alpha(e^x - 1) & , \quad x < 0 \end{cases}$$

En combinant tous les neurones d'une couche, un réseau à une seule couche peut approximer n'importe quelle fonction continue  $\hat{f}(x)$  sur un sous-ensemble compact de  $R_n$ , qui peut être formaté comme une combinaison linéaire de  $N$  neurones individuels comme suit :

$$\hat{f}(x) = \sum_{i=0}^{N-1} v_i h(w_i^T x + b_i)$$

Où  $v_i$  est le poids de la combinaison entre les neurones. Nous pouvons résumer tous les paramètres du réseau comme suit :

$$\theta = (v_0, b_0, w_0, \dots, v_N, v_N, v_N)^T$$

Afin d'augmenter la capacité du modèle et la capacité de représentation, nous pouvons introduire davantage de couches cachées dans le réseau entre la couche d'entrée et la couche de sortie, comme le montre la figure 2.1. Ces couches sont connectées entre elles pour former un réseau neuronal profond. Selon une théorie des réseaux neuronaux, les réseaux superficiels et profonds sont capables d'approximer arbitrairement toute fonction continue sur un domaine compact. Cependant, les réseaux profonds peuvent approximer la classe des fonctions de composition avec la même précision que les réseaux peu profonds, mais avec un nombre exponentiellement inférieur de paramètres d'apprentissage ainsi que de dimension VC. L'architecture profonde présente également des avantages pour la représentation des caractéristiques, comme la recombinaison des poids le long de différents chemins et la réutilisation des caractéristiques latentes.

## 2- Réseaux neuronaux convolutifs

Le réseau neuronal convolutif est un réseau neuronal profond spécialement conçu pour traiter les données d'image. Il a récemment été mis en évidence dans les tâches de vision par ordinateur, notamment la segmentation et la classification des images médicales. Il peut exploiter les informations spatiales et structurelles des images 2D ou 3D de son entrée et bénéficie des mécanismes du champ réceptif local, du partage de poids et du sous-échantillonnage.

Un réseau de classification d'images typique se compose de couches convolutionnelles, de couches de mise en commun, de couches entièrement connectées et d'une stratégie de saut de connexion. En plus de ces couches, le réseau de segmentation d'image typique comprend généralement aussi des couches de sous-échantillonnage pour sous-échantillonner la carte de caractéristiques afin de générer les cartes de segmentation finales.

Pour la classification d'images, les architectures les plus importantes sont AlexNet , VGGNet , Inception , ResNet et DenseNet.

Et, pour la segmentation d'image, les architectures de réseau populaires sont le réseau entièrement convolutif (FCN), U-Net, 3D U-Net/V-Net et leurs variantes.

Actuellement, les architectures ci-dessus sont largement utilisées comme pierre angulaire pour concevoir des solutions spécifiques. La figure illustre la structure d'un exemple de réseau neuronal convolutif de classification d'images.

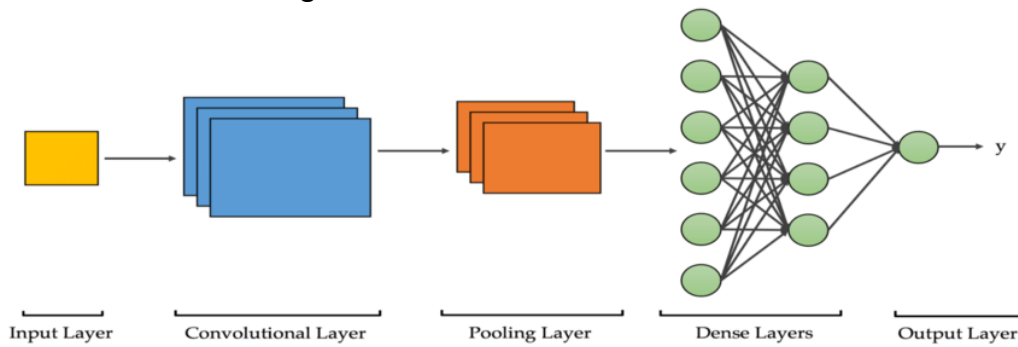


Figure 4 Réseaux de neurones

### 3- Couche convolutive

Les couches convolutives jouent un rôle central dans la construction des réseaux de neurones convolutifs.

La couche convolutive détecte les caractéristiques locales à des positions différentes de la couche précédente et mappe les informations apprises dans une nouvelle carte de caractéristiques. Pendant la convolution dans les réseaux, l'entrée de la couche précédente est divisée en perceptrons, créant des champs réceptifs locaux et compressant finalement les perceptrons en cartes de caractéristiques, qui indiquent les emplacements et la force d'une caractéristique détectée dans une entrée.

Dans la couche  $l$ , en supposant qu'il y ait un groupe de  $N^l$  filtres, chaque filtre détecte une caractéristique particulière à chaque position sur l'entrée. La sortie de la couche  $l$ , désignée par  $Y^{(l)}$ , sera constituée de  $N^l$  cartes de caractéristiques, où la  $i^{\text{ème}}$  carte de caractéristiques  $Y_i^{(l)}$  peut être calculée comme suit :

$$Y_i^{(l)} = h \left( \sum_{j=1}^{N^{(l-1)}} K_{i,j}^{(l)} * Y_j^{(l-1)} + B_i^{(l)} \right)$$

Où  $K_{i,j}^{(l)}$  est le noyau convolutif appliqué,  $B_i^{(l)}$  est une matrice de biais,  $h$  est la fonction d'activation illustrée précédemment.

### 4- Couche de regroupement ( Pooling Layer )

La couche de mise en commun dans le réseau convolutif vise à sous-échantillonner les cartes de caractéristiques afin de réduire le nombre de paramètres et le coût de calcul dans le réseau, ce qui est également bénéfique pour contrôler l'overfitting. Les opérations de mise en commun prennent en entrée une petite région d'une taille définie et produisent un seul nombre pour représenter cette région. La valeur représentative du champ réceptif est généralement calculée en employant la

fonction max appelée max-pooling ou la fonction moyenne appelée « average pooling ». Dans le réseau neuronal convolutif, une autre approche pour obtenir l'effet de déséchantillonnage de la mise en commun consiste à utiliser des convolutions avec un pas .

### 5- Couche de sur-échantillonnage ( Up-Sampling Layer )

Les couches de suréchantillonnage sont largement utilisées dans les réseaux de segmentation d'images pour suréchantillonner la carte de caractéristiques d'entrée à une résolution supérieure. La première approche est le rééchantillonnage et l'interpolation, qui consiste à redimensionner une carte de caractéristiques d'entrée à la taille souhaitée à l'aide d'une méthode d'interpolation telle que l'interpolation bilinéaire.

Une autre approche est le dégroupage, considéré comme l'inverse du regroupement (pooling).

Dans la couche d'unpooling, un inverse approximatif de la couche de pooling précédente est obtenu en enregistrant la position de chaque valeur d'activation maximale dans chaque région de pooling et cette information de position est ensuite utilisée pour placer les reconstructions de la couche précédente aux bons endroits.

La troisième approche est la convolution transposée. La convolution transposée est considérée comme l'inverse de l'opération de convolution, mais il ne s'agit pas d'une déconvolution mathématique correcte. Dans la convolution transposée, le noyau est placé au-dessus de l'entrée et les valeurs de l'entrée sont modifiées. l'entrée et les valeurs de l'entrée sont multipliées successivement par les poids du noyau pour produire le résultat échantillonné.

### 6- Sauter la connexion

Les sauts de connexions utilisés dans le réseau proposé introduisent des connexions qui sautent une ou plusieurs couches. Elles sont utiles pour simplifier l'optimisation d'un réseau.

Les modèles plus profonds ont tendance à rencontrer des obstacles pendant le processus de formation. Le signal du gradient disparaît lorsque la profondeur du réseau augmente. Mais les connexions de saut propagent le gradient à travers le modèle, ce qui peut atténuer le problème de disparition du gradient.

Dans les réseaux de segmentation avec une architecture d'encodeur-décodeur, les connexions de saut sont généralement utilisées entre chaque paire décodeur-encodeur, ce qui apporte les caractéristiques avec une résolution spatiale plus élevée des couches peu profondes de la partie encodeur directement aux couches de la partie décodeur pour la détection et la segmentation.

### 7- Loss function

La fonction de perte supervise la formation des réseaux. Dans la tâche de classification d'images, les fonctions de perte largement utilisées comprennent la perte d'entropie croisée catégorielle et la perte focale. La perte focale est une perte d'entropie croisée qui pondère la contribution de chaque échantillon à la perte en fonction de l'erreur de classification afin de résoudre le problème du déséquilibre des classes.

La perte d'entropie croisée catégorielle est définie comme suit :

$$L_{CE} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C \delta(y_i = c) \log(P(y_i = c))$$

où  $N$  désigne le nombre de données et  $C$  représente le nombre de catégories. ( $y_i = c$ ) est la fonction indicatrice et  $P(y_i = c)$  est la probabilité prédite par le modèle.

## Chapitre 3 : Extraction des données

Dans le cadre de ce projet, nous avons opté pour l'utilisation de la carte STM32 IoT Node et cela pour plusieurs raisons :

- La présence de capteurs embarqués sur la carte (Accéléromètre, Gyroscope et Microphones)
- La présence d'un microcontrôleur STM32L475 connu pour sa faible consommation donc adapté pour des applications liées à l'internet des objets.
- La présence de modules de communication sans fil pour des prochaines améliorations. (WIFI ou Bluetooth).

Nous allons utiliser deux approches différentes : l'une la détection des pannes à partir des données récoltées par le microphone, et l'autre par une combinaison de gyroscope et accéléromètre.

### 1- A partir du microphone

Le MP34DT01 est un microphone numérique ST-MEMS ultracompact, de faible puissance, omnidirectionnel, construit avec un élément de détection capacitif et une interface IC.

L'élément de détection, capable de détecter les ondes acoustiques, est fabriqué à l'aide d'un procédé spécialisé de micro-usinage du silicium dédié à la production de capteurs audio.

L'interface IC est fabriquée à l'aide d'un procédé CMOS qui permet de concevoir un circuit dédié capable de fournir un signal numérique externe au format PDM.

La PDM est une forme de modulation utilisée pour représenter un signal analogique dans le domaine numérique. Il s'agit d'un flux haute fréquence d'échantillons numériques de 1 bit. Dans un signal PDM, la densité relative des impulsions correspond à l'amplitude du signal analogique. Un grand nombre de 1 correspond à une valeur d'amplitude élevée (positive), alors qu'un grand nombre de 0 correspondrait à une valeur d'amplitude faible (négative), et l'alternance de 1 et de 0 correspondrait à une valeur d'amplitude nulle.

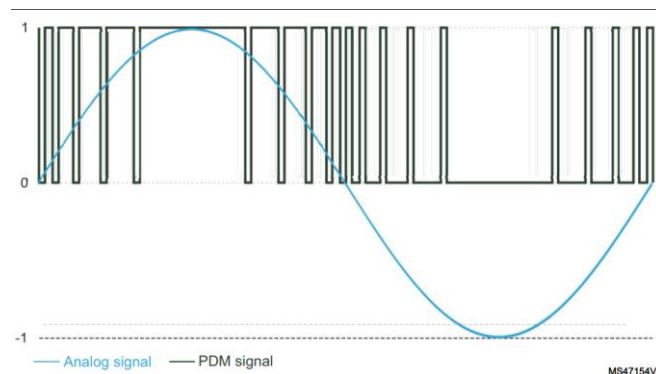


Figure 5 Signal d'un microphone PDM

Dans le signal PCM, des valeurs d'amplitude spécifiques sont codées en impulsions.

Un flux PCM possède deux propriétés de base qui déterminent la fidélité du flux au signal analogique d'origine :

- la fréquence d'échantillonnage
- la profondeur de bits

La fréquence d'échantillonnage est le nombre d'échantillons d'un signal pris par seconde pour le représenter numériquement. La profondeur binaire détermine le nombre de bits d'information dans chaque échantillon.

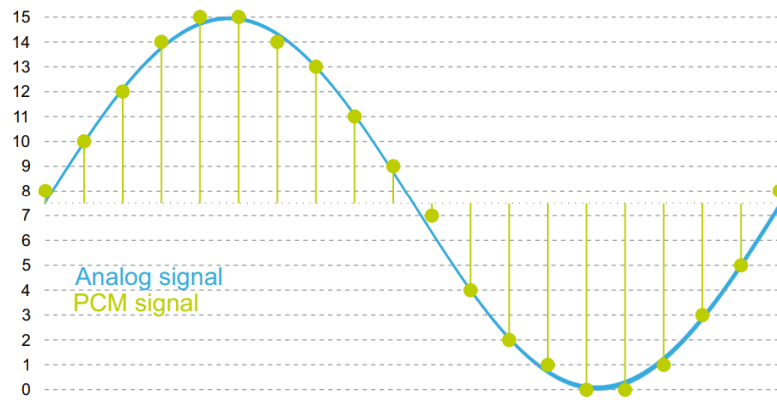


Figure 6 L'échantillonnage du flux PDM en échantillons PCM

Afin de convertir le flux PDM en échantillons PCM, le flux PDM doit être filtré et décimé. Dans l'étape de décimation, la fréquence d'échantillonnage du signal PDM est réduite à la fréquence d'échantillonnage audio visée (16 kHz par exemple). En sélectionnant 1 échantillon sur M, la fréquence d'échantillonnage est réduite d'un facteur M.

Par conséquent, la fréquence des données PDM (qui est la fréquence de l'horloge du microphone) est égale à M fois la fréquence d'échantillonnage audio cible nécessaire dans une application, où M est la fréquence de décimation.

$$\text{Fréquence PDM} = \text{Fréquence d'échantillonnage audio} \times \text{facteur de décimation}$$

Une fois les données extraites nous allons calculer le Log-Mel Spectrogramme.

Pour faire court, le log-mel spectrogram est comme un ensemble de FFT mis bout à bout tout en appliquant une modification d'échelle pour se rapprocher du comportement de l'oreille humaine.

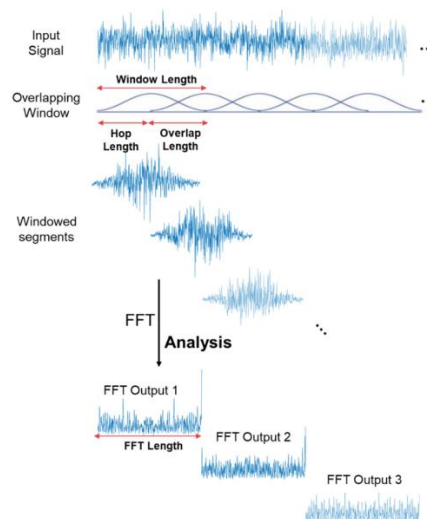


Figure 7 Calcul du Log-Mel Spectrogramme

## 2- A partir de l'accéléromètre et du gyroscope

Le LSM6DSL est un système intégré comprenant un accéléromètre numérique 3D et un gyroscope numérique 3D, fonctionnant à 0,65 mA en mode haute performance et permettant des fonctions de faible consommation.

Le LSM6DSL a été conçu pour mettre en œuvre des fonctions telles que le mouvement significatif, l'inclinaison, fonctions de podomètre, de détecteur et de compteur de pas, d'horodatage et de prise en charge de l'acquisition de données d'un magnétomètre externe.

Le LSM6DSL dispose d'une plage d'accélération pleine échelle de  $\pm 2/\pm 4/\pm 8/\pm 16$  g et d'une plage de fréquence angulaire de  $\pm 125/\pm 245/\pm 500/\pm 1000/\pm 2000$  dps.

Les registres intégrés dans le LSM6DSL sont accessibles via les interfaces série I2C et SPI. Sur le STM32L4 Discovery kit for IoT node, le bus I2C2 du STM32L475VGT6 est utilisé.

Un accéléromètre est un dispositif électromécanique utilisé pour mesurer les forces d'accélération. Ces forces peuvent être :

- Statique, comme la force de gravité
- Dynamiques comme les forces de mouvement ou de vibration ce qui permet la détection de mouvement

L'accéléromètre retourne les composantes de la force d'inertie qui est dirigée dans la direction opposée au vecteur d'accélération.

Un gyroscope est un dispositif utilisé pour mesurer ou maintenir l'orientation et la vitesse angulaire. Il s'agit d'une roue tournante, ou d'un disque, dans lequel l'axe de rotation est libre de prendre toute orientation par lui-même. Lors de la rotation, l'orientation de cet axe n'est pas affectée par l'inclinaison ou la rotation du support, en fonction de la conservation du moment angulaire.

## Chapitre 4 : Etude pratique et implémentation

### Problématique

Les ventilateurs sont souvent utilisés dans des milieux industriels et cela dans plusieurs buts que ce soit l'aération ou le refroidissement. De ce fait, ils représentent un maillon critique de toute la chaîne d'où le besoin de détecter les défaillances le plus tôt possible.

### Le but du projet

Il est question d'exploiter une des différentes pistes pour détecter les défaillances des ventilateurs dans un milieu industriel. Dans notre cas, cela sera la détection de différentes anomalies ( un déséquilibre des ailes , le colmatage , un changement de tension) à travers le son produit par le ventilateur et par la vibration du ventilateur.

### Cadre général de travail

Le projet peut être divisé en plusieurs étapes :

- Le téléchargement du dataset et son préprocessing
- La recherche de modèle de Deep Learning adéquat et son entraînement
- Extraction des données sur STM32\$
- Importation du modèle sur microcontrôleur et le mettre en marche.

### 1- Approche à base du son

#### a- Présentation du dataset

Le jeu de données a été collecté à l'aide du microphone TAMAGO-03, fabriqué par System In Frontier Inc. Il s'agit d'un réseau de microphones circulaire composé de huit microphones distincts. L'utilisation du réseau de microphones permet d'évaluer non seulement les approches basées sur un seul canal, mais aussi celles basées sur plusieurs canaux. Le réseau de microphones a été maintenu à une distance de 50 cm de la machine ; des segments sonores de 10 secondes ont été enregistrés. L'ensemble de données contient huit canaux séparés pour chaque segment. Il convient de noter que chaque son de machine a été enregistré dans une session séparée. En condition de fonctionnement, le son de la machine a été enregistré sous forme de signaux audio 16 bits échantillonnés à 16 kHz dans un environnement réverbérant. En outre le son de la machine cible, le bruit de fond de plusieurs usines réelles a été enregistré en continu afin de le mélanger au son de la machine cible pour simuler des environnements réels. Pour enregistrer le bruit de fond, nous avons utilisé le même réseau de microphones que pour le son de la machine cible. de microphones que pour le son de la machine cible. L'ensemble de données MIMII contient le son de quatre types de machines différentes : valves, pompes, ventilateurs et glissières. Les ventilateurs représentent des ventilateurs industriels, qui sont utilisés pour fournir un flux ou un gaz d'air continu dans les usines. Les types de sons produits par les machines sont stationnaires et non stationnaires, ont différentes caractéristiques et différents degrés de difficulté. Chaque type de machine est composé de sept machines individuelles. Les machines individuelles peuvent être d'un modèle de produit différent. Le nombre de segments sonores pour chaque son anormal pour chaque type de machine est faible car nous considérons la cible principale de notre ensemble de données comme un scénario

d'apprentissage non supervisé. scénario d'apprentissage non supervisé et considérons les segments anormaux comme une partie des données de test.

Machine ID	Segments de condition normal	Segments de condition anormal
00	1011	407
01	1034	407
02	1016	359
03	1012	358
04	1033	348
05	1109	349
06	1015	361

#### b- Préprocessing

Pour créer un spectrogramme mel à partir d'ondes audio, nous utiliserons la bibliothèque librosa. C'est un package python pour l'analyse musicale et audio. Il fournit les blocs de construction nécessaires pour créer des systèmes de recherche d'informations musicales.

Les résultats obtenus sur un exemple des fichiers son du dataset original:

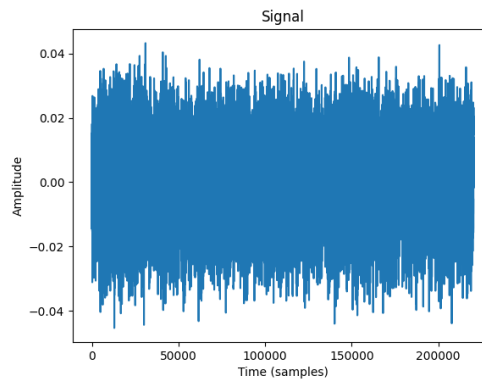


Figure 8 La forme originale des ondes sonores

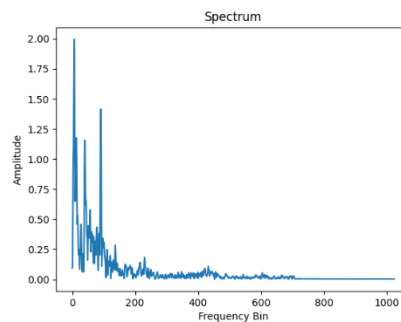


Figure 9 Spectre obtenue par transformation fréquentielle

Le préprocessing consiste à préparer un fichier csv contenant la métadonnée. De plus, il est question de normaliser tous les fichiers présents (même durée, même fréquence d'échantillonnage ainsi qu'un format mono) pour que les spectrogrammes obtenus soient d'une même dimension (dans notre cas 30\*32) comme le montre la figure suivante :



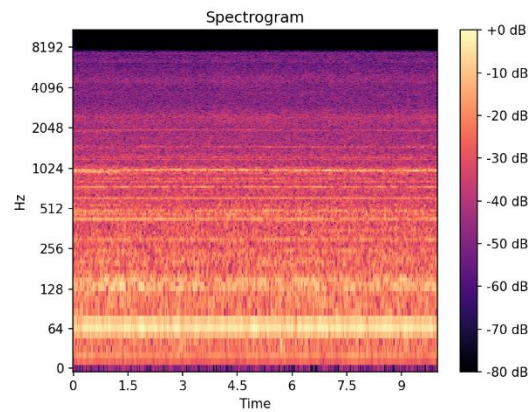


Figure 10 Le Mel-Spectrogramme original

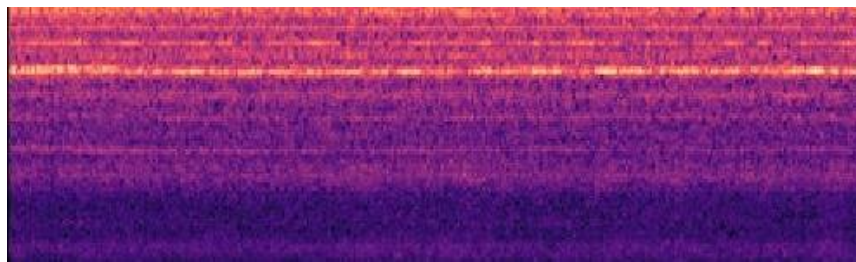


Figure 11 Le Spectrogramme normalisé

### c- Choix du modèle

Le choix du modèle se fait sur plusieurs critères :

- Le format des données d'entrée
- Le format des données de sortie (le nombre de classe) dans notre cas 2 possibilités
- Le nombre de couches et de neurones de sorte que cela ne dépasse pas la RAM et mémoire Flash sur STM32

Les modifications sont faites à partir d'un modèle de classification de son prédéfini pour aboutir au résultat suivant :

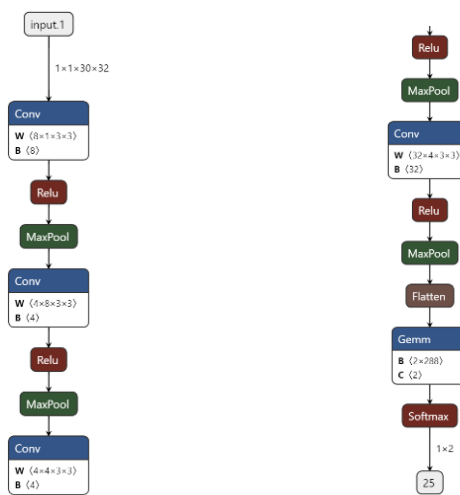


Figure 12 Le Modèle de classification pour la première approche

### d- Implémentation sur STM32

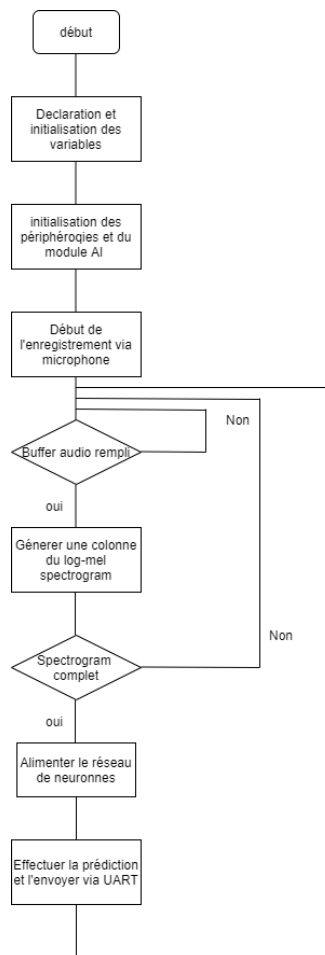


Figure 13 Logigramme d'implémentation sur STM32 pour la première approche

Pour que nos résultats sur microcontrôleur soit le plus fiable possible, on doit suivre le même cheminement que sur PC. Comme le montre le logigramme ci-dessus, on commencera par enregistrer le son à travers le microphone.

Une fois le tableau des échantillons est sauvegardé, on se chargera de calculer sur microcontrôleur le log-mel Spectrogramme pour ensuite l'assigner comme entrée à notre réseau de neurones.

#### e- Résultat

```

mized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/20
2360/2360 [=====] - 56s 24ms/step - loss: 0.7214 - Accuracy: 0.5119
Epoch 2/20
2360/2360 [=====] - 44s 19ms/step - loss: 0.6972 - Accuracy: 0.5025
Epoch 3/20
2360/2360 [=====] - 41s 17ms/step - loss: 0.6964 - Accuracy: 0.4919
Epoch 4/20
2360/2360 [=====] - 41s 18ms/step - loss: 0.6952 - Accuracy: 0.4970
Epoch 5/20
2360/2360 [=====] - 41s 18ms/step - loss: 0.6946 - Accuracy: 0.4987
Epoch 6/20
2360/2360 [=====] - 42s 18ms/step - loss: 0.6942 - Accuracy: 0.4919
Epoch 7/20
2360/2360 [=====] - 43s 18ms/step - loss: 0.6941 - Accuracy: 0.4852
Epoch 8/20
2360/2360 [=====] - 45s 19ms/step - loss: 0.6938 - Accuracy: 0.4881
Epoch 9/20
2360/2360 [=====] - 46s 20ms/step - loss: 0.6938 - Accuracy: 0.4983
Epoch 10/20
2360/2360 [=====] - 42s 18ms/step - loss: 0.6937 - Accuracy: 0.5072
Epoch 11/20
2360/2360 [=====] - 45s 19ms/step - loss: 0.6936 - Accuracy: 0.4966
Epoch 12/20
2360/2360 [=====] - 41s 18ms/step - loss: 0.6936 - Accuracy: 0.4987
Epoch 13/20
2360/2360 [=====] - 40s 17ms/step - loss: 0.6930 - Accuracy: 0.5127
Epoch 14/20
2360/2360 [=====] - 44s 19ms/step - loss: 0.6937 - Accuracy: 0.4869
  
```

Figure 14 Le résultat obtenu

Le modèle fonctionne effectivement sur STM32 selon le logigramme présenté ci-dessus. Cependant présente un rendu peut fiable et cela pour plusieurs raisons :

- Le dataset utilisé a été enregistré avec un autre microphone que celui présent sur la carte
- Un dataset plus riche peut être une solution
- La faible résolution du log-mel spectrogramme est obligatoire dû à la faible mémoire du microcontrôleur du coup il est possible d'envisager comme solution l'augmentation de la résolution en cas d'utilisation de meilleur MCU

#### f- Amélioration du modèle :

Pour améliorer les performances de notre modèle et pour la même architecture de réseau, on lui applique des modifications:

#### 1- On divise le dataset en 3 ensembles: Train, Validation, Test



Figure 15 La division du dataset

L'ensemble de validation est utilisé pour évaluer le modèle, mais il s'agit d'une évaluation fréquente pour affiner les hyperparamètres du modèle. Nous utilisons les résultats de l'ensemble de validation afin de décider quand est-ce qu'il faut arrêter l'apprentissage. Ceci est possible en détectant le point "sweet spot" ou "point optimum" pour lequel les résultats de l'entraînement sont optimisés. Même si le modèle ne termine pas le nombre d'époques précisé par le code, l'entraînement est interrompu après un certain nombre d'essais qui indiquent qu'on est arrivé au sweet spot.

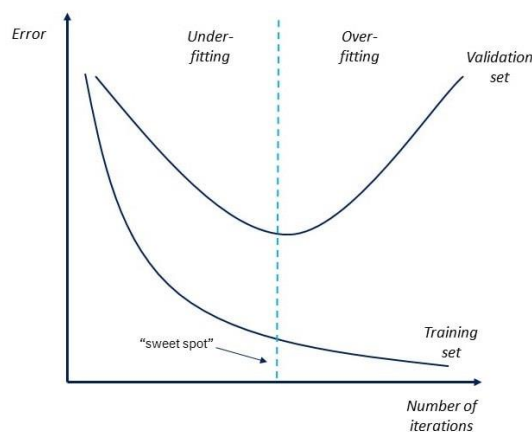


Figure 16 La relation entre le nombre d'iterations et la précision du modèle

À propos du taux de fractionnement de l'ensemble de données

Diviser l'ensemble de données en ensembles d'apprentissage, de validation et de test dépend principalement de 2 choses. Premièrement, le nombre total d'échantillons dans les données (2950 images) et deuxièmement, sur le modèle réel que vous entraînez.

Les modèles avec très peu d'hyperparamètres seront faciles à valider et à ajuster, vous pouvez donc probablement réduire la taille de votre ensemble de validation, mais si votre modèle a de nombreux hyperparamètres, vous voudrez également avoir un grand ensemble de validation. D'où dans notre cas 10% de l'ensemble des données est suffisant. Les hyperparamètres doivent être équilibrément répartis entre les données de validation et de test.

## 2- On change les valeurs de arguments de la fonction fit

Le batch\_size qui été à 1 sera batch\_size=16, # Nombre d'échantillons par mise à jour du gradient.

On augmente le nombre d'epochs epochs=1000, # Nombre d'époques pour former le modèle. Une époque est une itération sur l'ensemble des données x et y fournies

Même avec le nombre d'epochs très grand, l'entraînement ne continue pas jusqu'à 1000 epochs puisqu'on lui demande de s'arrêter en fonction de la valeur de la "validation accuracy"

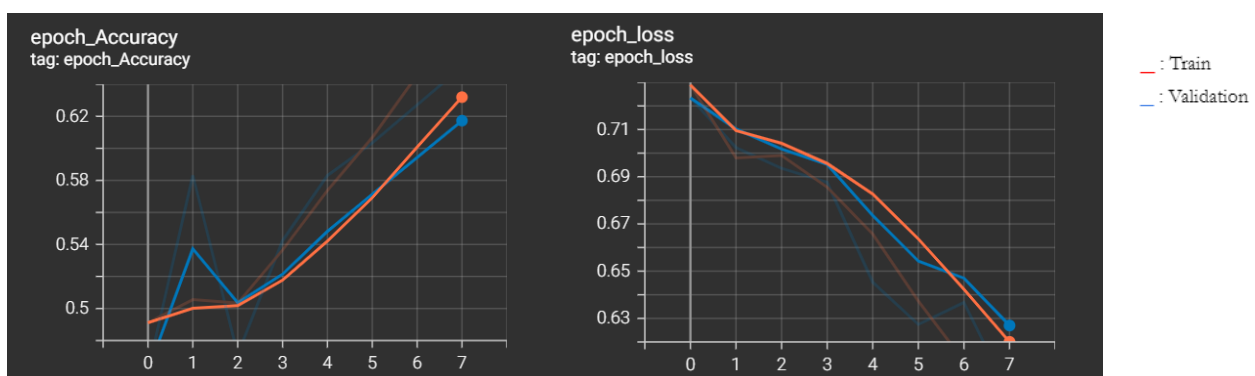


Figure 17 Visualisation de l'évolution de l'entraînement

## g- Résultat

On peut bien remarquer que les performances sont améliorées et que les données d'entraînement et de validation nous permettent d'obtenir une meilleure "Accuracy" qui va jusqu'à 80% et une meilleure "epoch loss" qui est bien plus faible que la première simulation.

```
- val_Accuracy: 0.7898
Epoch 19/1000
148/148 [=====] - 27s 179ms/step - loss: 0.3638 - Accuracy: 0.8318 - val_loss: 0.4632
- val_Accuracy: 0.7492
Epoch 20/1000
148/148 [=====] - 27s 184ms/step - loss: 0.3495 - Accuracy: 0.8479 - val_loss: 0.4426
- val_Accuracy: 0.7797
Epoch 21/1000
148/148 [=====] - 29s 196ms/step - loss: 0.3386 - Accuracy: 0.8445 - val_loss: 0.3801
- val_Accuracy: 0.8136
Epoch 22/1000
148/148 [=====] - 36s 240ms/step - loss: 0.3178 - Accuracy: 0.8576 - val_loss: 0.4526
- val_Accuracy: 0.7864
```

Figure 18 Résultat obtenu après amélioration

On remarque que l'entraînement s'achève bien avant la fin des 1000 epochs.

L'ensemble de données de test n'est utilisé qu'une fois qu'un modèle est complètement formé.

Enfin, on veut voir dans quelle catégorie le modèle mettrait une forme son "Abnormal".

```
DCN model prediction: [['Abnormal']]

Probabilities for each category:
Abnormal : 0.53782207
Normal : 0.43066946
```

Figure 19 Résultat du test

Notre modèle a bien réussi à trouver le bon label à notre spectrogramme.

## 2- Approche à base de la vibration

### a- Présentation du Dataset

Cet ensemble de données est destiné à accompagner un kit de démonstration matériel disponible auprès de STMicroelectronics pour montrer le potentiel de la solution STMicroelectronics / SensiML pour construire rapidement des algorithmes de classification de données de capteurs intelligents en série temporelle afin de traduire les données brutes de capteurs physiques en événements de compréhension spécifiques à l'application en utilisant la reconnaissance de formes.

Ces événements comprennent le choc des pales, le déséquilibre du moyeu et le choc du châssis. La classification de l'état du moteur comprend ces événements ainsi que la détection des vibrations des états de marche et d'arrêt du moteur.

Toutes les classifications des événements de ventilateur sont basées sur la surveillance des vibrations/mouvements détectés par l'unité de mesure inertielle embarquée de la STBox (accéléromètre 6 axes et capteur gyroscopique).

De cette façon, la démonstration simule un cas d'utilisation commun souhaité dans les applications de maintenance prédictive industrielle pour un système de capteurs qui peut détecter l'utilisation de l'équipement, les états de défaillance et la dégradation des performances avec une incursion minimale ou nulle dans les systèmes électromécaniques existants utilisés pour le contrôle et le fonctionnement.

L'ensemble de données est constitué de sessions capturées de valeurs brutes de capteurs IMU échantillonnées à 416 Hz. Chaque session contient des segments étiquetés d'un type d'état/classification de ventilateur donné parmi la carte de classe suivante d'événements à reconnaître :

Numéro	Nom	Description
0	Unknown	Mouvement Inconnu
1	Blade Fault	Obstruction des lames
2	Flow Blocked	L'entrée de l'air est bloquée
3	Off	Ventilateur en état d'arrêt
4	On	Ventilateur en état de marche
5	Guard Tamper	Quelque chose est en train de toucher la garde du ventilateur
6	Mount Fault	Cadre du ventilateur instable
7	Tapping	Quelque chose est en train de toucher la partie supérieure du ventilateur

### b- Préprocessing

Vu la forme du dataset actuel, on se propose de changer la forme de ce dernier.

On utilisera un script python, permettant de parcourir tous les fichiers csv , extraire les données sous forme de dataframe et ajouter une colonne indiquant le label ( dans notre cas l'état du ventilateur industriel ) ensuite on exportera sous forme de CSV pour faciliter la manipulation des fichiers.

On remarque que le dataset n'est pas équilibré, d'où la nécessité de rééquilibrage pour un model plus précis. Il y a risque d'overfit pour un état bien défini de ventilateur.

### c- Choix du modèle

Le choix du modèle s'est fait sous les mêmes contraintes que pour la première méthode. Seulement, les dimensions des entrées change et pour les sorties de même (plus précisément on est face à 7 classes)

On s'est basé sur un modèle de reconnaissance d'activité humaine qu'on a adapté à nos besoins pour aboutir au résultat suivant :

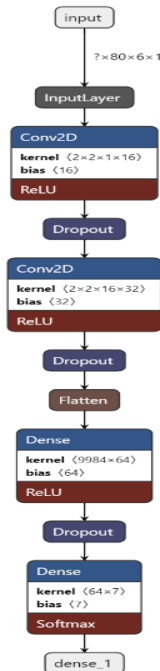


Figure 20 Le Modèle de classification pour la deuxième approche

### d- Implémentation sur STM32

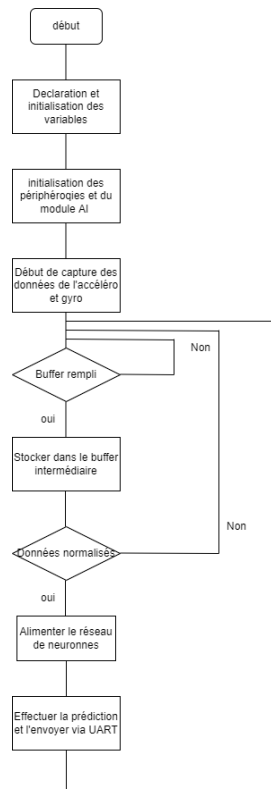
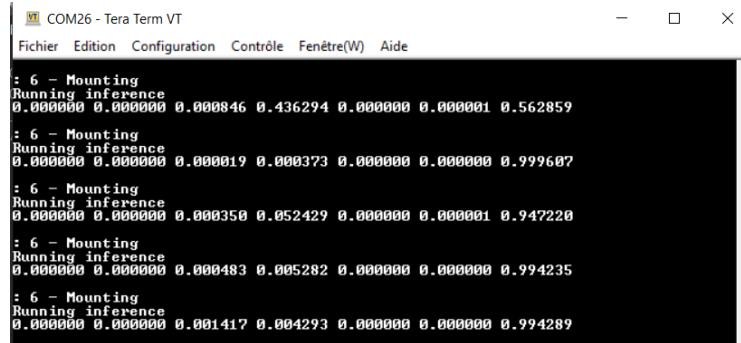


Figure 21 Logigramme d'implémentation sur STM32 pour la deuxième approche

Comme le montre la figure ci-dessus, on commencera par initialiser les variables ainsi que les périphériques utilisés (UART, GPIO, I2C et Timer ). Ensuite on se chargera de stocker par groupe de 6 données à chaque itération du Callback du Timer pour aboutir à remplir un tableau de 80\*6 pour pouvoir le fournir comme entrée au réseau de neurones implémenté.

#### e- Résultat



```
COM26 - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide

: 6 - Mounting
Running inference
0.000000 0.000000 0.000846 0.436294 0.000000 0.000001 0.562859

: 6 - Mounting
Running inference
0.000000 0.000000 0.000819 0.000373 0.000000 0.000000 0.999607

: 6 - Mounting
Running inference
0.000000 0.000000 0.000350 0.052429 0.000000 0.000001 0.947220

: 6 - Mounting
Running inference
0.000000 0.000000 0.000483 0.005282 0.000000 0.000000 0.994235

: 6 - Mounting
Running inference
0.000000 0.000000 0.001417 0.004293 0.000000 0.000000 0.994289
```

Figure 22 Le resultat obtenu

Le résultat est assez fiable en théorie sur PC mais assez limité sur microcontrôleur et cela pour les raisons purement mécaniques. Le dataset utilisé a été conçu pour un autre kit de développement que l'IoT node et donc mécaniquement incompatible avec notre carte utilisée.

## Conclusion

Le travail expliqué brièvement à travers les dernières pages a été planifié, réalisé et déployé avec succès. Cependant, le projet comporte des points sur lequel on peut toujours travailler afin de l'améliorer et de le rendre potentiellement commercialisable :

- Le dataset utilisé peut provoquer des limitations quant au type de ventilateurs utilisés. Par conséquent, il est envisageable dans un certain temps de créer un dataset propre à l'installation industrielle et l'environnement de travail du potentiel client
- Dans le cas idéal, une alimentation externe devient obligatoire ainsi que des modifications du programme pour limiter la consommation d'énergie.
- Le modèle de deep learning présente une précision limitée étant donné le temps de développement relativement court du projet. Ainsi ce point doit être amélioré.



## Références :

- *AI : Artificial Intelligence overview - stm32mcu.* (s. d.). Wiki.ST. Consulté le 8 septembre 2021, à l'adresse [https://wiki.st.com/stm32mcu/wiki/AI:Artificial\\_Intelligence\\_overview](https://wiki.st.com/stm32mcu/wiki/AI:Artificial_Intelligence_overview)
- Purohit, H., Tanabe, R., Ichige, T., Endo, T., Nikaido, Y., Suefusa, K., & Kawaguchi, Y. (2019). MIMII Dataset : Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. Published. <https://doi.org/10.33682/m76f-d618>
- Roberts, L. (2020, 14 mars). *Understanding the Mel Spectrogram - Analytics Vidhya*. Medium. <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>
- *Running on Windows : Node-RED.* (s. d.). Node-RED. Consulté le 8 septembre 2021, à l'adresse <https://nodered.org/docs/getting-started/windows>
- STMicroelectronics. (s. d.-a). *B-L475E-IOT01A*. Consulté le 8 septembre 2021, à l'adresse <https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html>
- STMicroelectronics. (s. d.-b). *STM32StepByStep : Step4 Sensors usage - stm32mcu*. Wiki.ST. Consulté le 8 septembre 2021, à l'adresse [https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:Step4\\_Sensors\\_usage](https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:Step4_Sensors_usage)
- Sura, H. (2020, 20 janvier). *Audio Classification using Librosa and Pytorch - Hasith Sura*. Medium. <https://medium.com/@hasithsura/audio-classification-d37a82d6715>