TP 2 – TinyML – IA embarquee

Objectif

Découverte du TinyML au travers :

- De premières expérimentations utilisant un smartphone
- D'une exploitation des possibilités du TinyML utilisant une carte électronique (Wio terminal dans ce TP)

Les implémentations qui sont proposées dans ce TP utilisent Edge Impulse – actuellement principale plateforme de développement pour l'apprentissage automatique sur les dispositifs embaqués.

TinyML – Introduction

TinyML désigne l'implémentation de modèles d'apprentissage automatique (Machine Learning) sur des dispositifs embarqués à faible consommation d'énergie, permettant l'exécution de tâches d'IA directement sur des appareils tels que des microcontrôleurs.

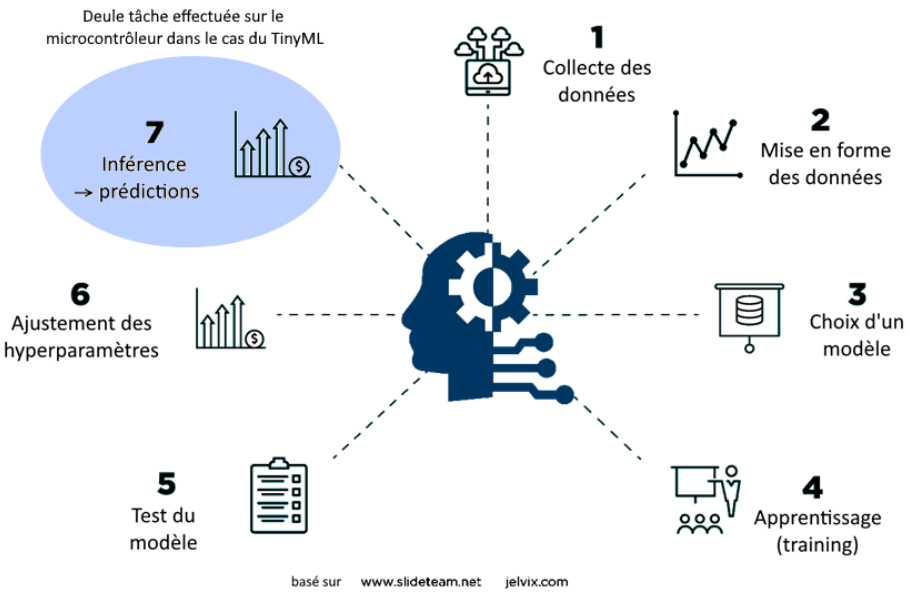
Avantages du TinyML :

- Faible consommation d'énergie.
- Latence réduite : Les calculs locaux éliminent le besoin de transmission vers le cloud.
- Coût réduit : Fonctionne sur du matériel peu coûteux comme les microcontrôleurs.
- Confidentialité accrue : Les données sont traitées localement, minimisant les risques de fuite.
- Portabilité : Permet l'intégration dans de petits appareils non connectés ou dans des endroits isolés.

Inconvénients du TinyML :

- Capacité limitée : Les ressources en mémoire et en puissance de calcul sont restreintes.
- Performances moindres : Les modèles doivent souvent être simplifiés, ce qui peut réduire leur précision.
- Domaine émergent : Moins de standardisation et d'outils matures par rapport à l'IA traditionnelle (pour le moment).

Dans le cadre du TinyML, la phase d'apprentissage est gérée par un ordinateur (distant ou non) car le microcontrôleur sur lequel on souhaite au final utiliser le modèle d'IA n'est pas assez puissant pour effectuer les calculs. Le modèle est ensuite compilé puis utilisé au sein d'une bibliothèque ou d'une application sur le microcontrôleur ou le smartphone (voir zone colorée sur le schéma ci-contre).



Remarque : Edge Impulse est aujourd'hui utilisé dans de plus en plus d'entreprises. Nous utiliserons la version gratuite, limitée, dans ce TP.

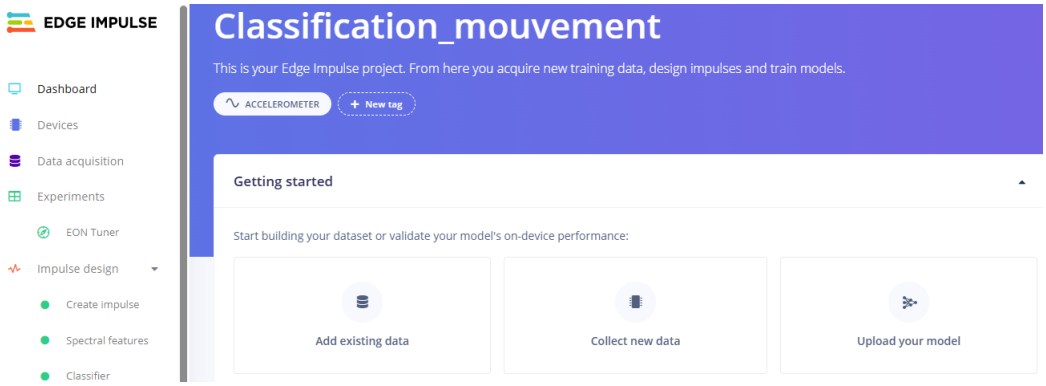
PARTIE 1 : DEPLOIEMENT SUR UN SMARTPHONE

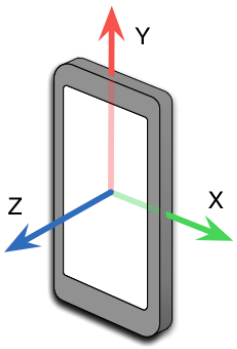
Les exemples de cette partie sont basés sur le travail de Louis Moreau
(Responsable des relations avec les développeurs chez Edge Impulse) publié sur Github.

1.1 CLASSIFICATION DE MOUVEMENT

Q1. *Imaginer 3 situations dans lesquelles la classification de mouvement sur un smartphone pourrait être utile.*

- Depuis un ordinateur, aller sur le site de Edge Impulse. Créer un compte et s'y connecter.
- Créer un nouveau projet en cliquant sur l'icone ronde de votre profil en haut à droite de l'écran. Nommer le projet "Classification_mouvement".



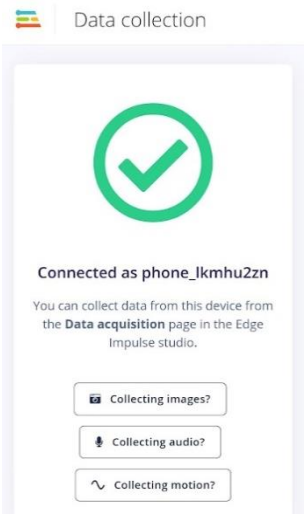


Nous allons ici résoudre un problème de classification associé à des mouvements du smartphone. Pour cela, commencer par identifier clairement 3 mouvements (qui seront 3 classes) que vous souhaiterez distinguer. Cela peut être par exemple :

- Secouer l'appareil verticalement
- Secouer l'appareil horizontalement
- Retourner l'appareil

1.1.1 COLLECTE DES DONNEES

Nous allons pour commencer acquérir des données qui serviront plus tard à entrainer le modèle d'IA. Ces données seront acquises grâce à la centrale inertielle (accéléromètre + gyroscope) de votre smartphone. L'ensemble des données acquises sera nommé dataset.



- Depuis le Dashboard du projet, cliquer sur "Collect new data". Une fenêtre s'ouvre et présente 3 méthodes d'acquisition de données.
- Scanner le QR Code qui s'affiche avec votre smartphone. Cela ouvre une page web permettant l'acquisition des données (Figure ci-contre).
- Cliquer sur l'ordinateur sur "Get started". On arrive alors dans la fenêtre d'acquisition des données. Le smartphone apparaît comme connecté sur la droite de la fenêtre.

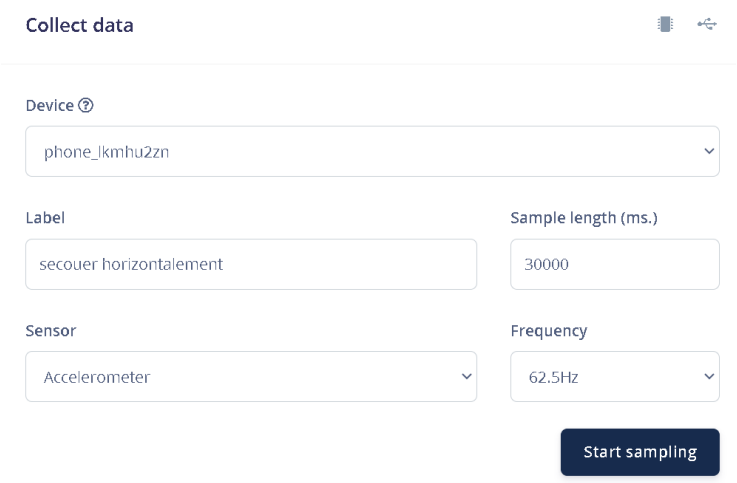
Nous allons maintenant effectuer l'acquisition des données qui serviront à entrainer puis à tester l'algorithme d'IA. On rappelle que **l'apprentissage automatique nécessite une grande quantité de données** pour mener à de bons résultats.

On insistera également sur la **diversité parmi les classes** : Par exemple, pour "secouer l'appareil verticalement", on fera le mouvement à différentes vitesses, et autour de différents axes.

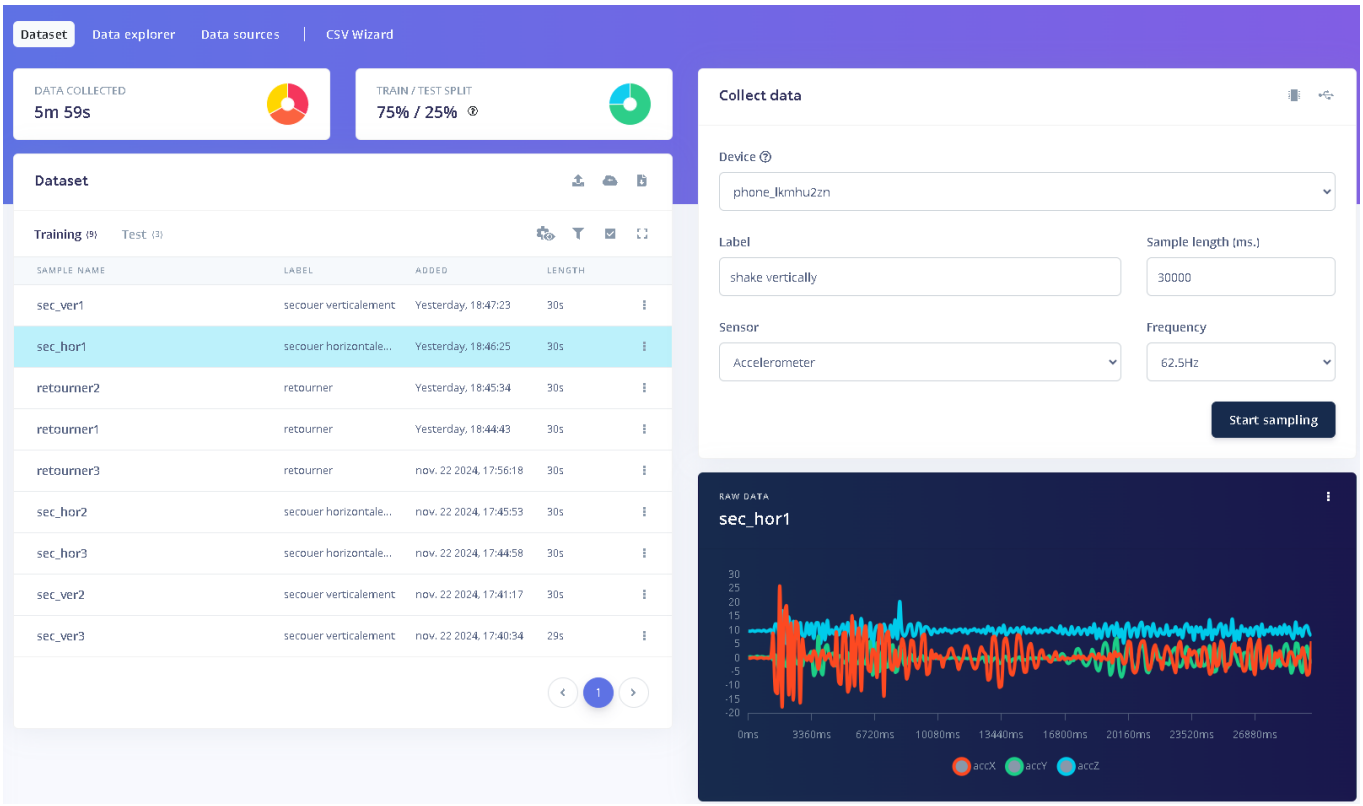
- Dans la fenêtre d'acquisition des données (Collect data – figure de gauche page suivante), choisir une durée d'acquisition de 30 s = 30 000 ms par exemple. Choisir l'accéléromètre du smartphone comme

capteur à enregistrer. Donner un label = nom à une première classe de mouvement, par exemple "secouer verticalement".

- Cliquer sur start sampling sur l'ordinateur : l'acquisition du signal démarre automatiquement au bout de quelques secondes sur le smartphone (figure de droite page suivante). Il faut alors effectuer et renouveler le mouvement en faisant de légères variations (vitesse, amplitude, orientation…) jusqu'à la fin de l'acquisition



- A la fin de l'acquisition, l'échantillon (la mesure effectuée) apparaît sur l'ordinateur dans le tableau Dataset. En cliquant dessus, on visualise sur une courbe l'évolution temporelle des accélérations mesurées suivant les trois directions (Figure ci-dessous).

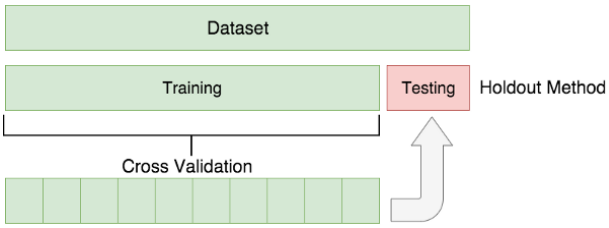


- Effectuer des mesures en utilisant la méthode précédente pour chacun des 3 mouvements définis précédemment. On effectuera plusieurs mesures de manière à avoir au final une durée d'enregistrement d'environ 2 minutes (ou plus) pour chaque classe à identifier (4 acquisitions de 30 s par classe).

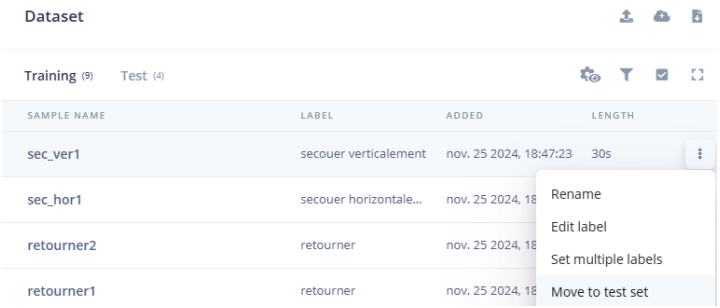
Q2. *Observer les allures des courbes d'accélération obtenues pour chacune des trois classes. Notez-vous des caractéristiques distinguant chacune des classes ? Observez-vous également de la diversité parmi les classes (variations de l'allure des signaux au sein d'une même classe) ?*

Nous allons maintenant séparer le dataset en 2 ensembles :

- Données d'entrainement (environ 80% du dataset) → Entrainement du modèle d'IA
- Données de test (environ 20% du dataset) → Test des performances du modèle



Source : https://medium.com



"Move to test set" dans Edge Impulse

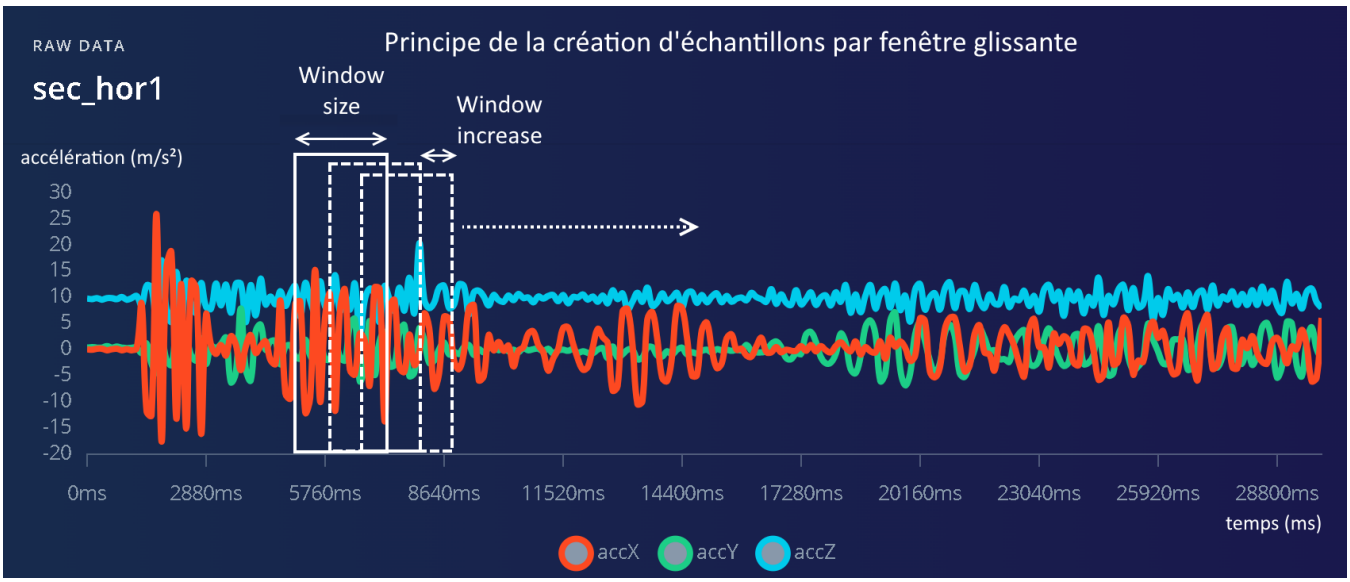
- Pour un des échantillons de chaque classe : cliquer sur les trois points situés à droite de l'échantillon (écran représenté sur la figure de la page précédente), puis sélectionner "move to test set". On obtient alors une répartition données d'entrainement / de test de 75% / 25% qui est couramment utilisée.

1.1.2 Mise en forme des donnees

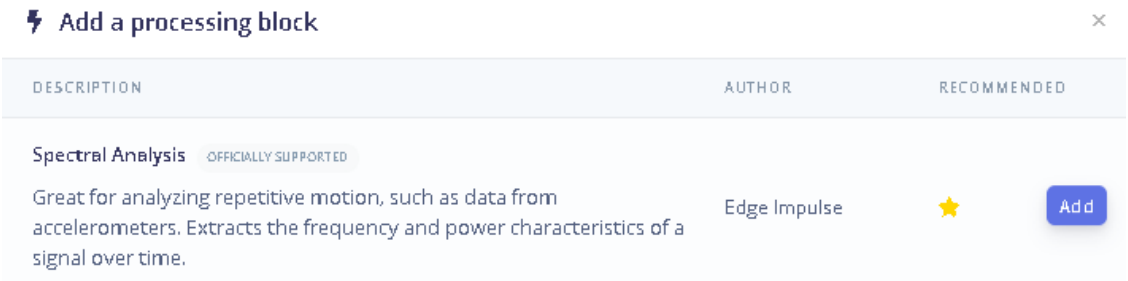
Avant d'entrainer l'algorithme d'IA à reconnaitre les différentes classes, il faut extraire des données temporelles acquises des attributs (features en anglais), qu'on peut voir comme des "caractéristiques principales" des signaux.

Des différences parmi ces attributs permettront de distinguer et de classer les différents mouvements.

- Dans l'arborescence de gauche, dans la rubrique Impulse design, cliquer sur "Create impulse" et valider les options de déploiement par défaut.
- Configurer le bloc "Time series data" pour que la valeur "Window size" corresponde à une durée incluant l'ensemble d'un mouvement à reconnaître : ici 2s = 2000 ms devrait convenir. A partir des échantillons acquis précédemment (durée 30s), Edge Impulse va créer plusieurs échantillons d'une durée correspondant à "Window size" (ici 2s). Pour cela, il utilise la méthode de la fenêtre glissante (sliding window) présenté ci-dessous : on déplace une fenêtre d'une durée de 2s par incréments de la valeur "window increase" (200 ms par défaut). On crée ainsi un grand nombre d'échantillons à partir d'une seule acquisition.

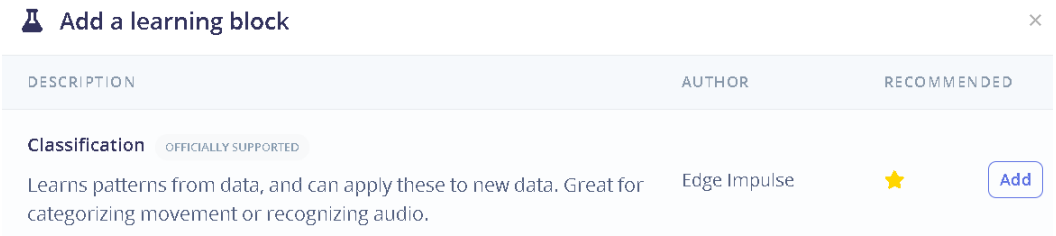


- Cliquer sur "Add a processing block" et sélectionner "Spectral analysis". Une courte description de l'objectif de ce bloc de traitement est donnée. On y note que l'analyse spectrale est adaptée à l'analyse de données temporelles issues d'accéléromètres, ce qui est le cas dans l'exemple traité. Conserver dans ce bloc d'analyse spectrale l'analyse des accélérations mesurées suivant les 3 directions. Ce bloc permettra d'extraire des attributs caractéristiques des échantillons mesurés.



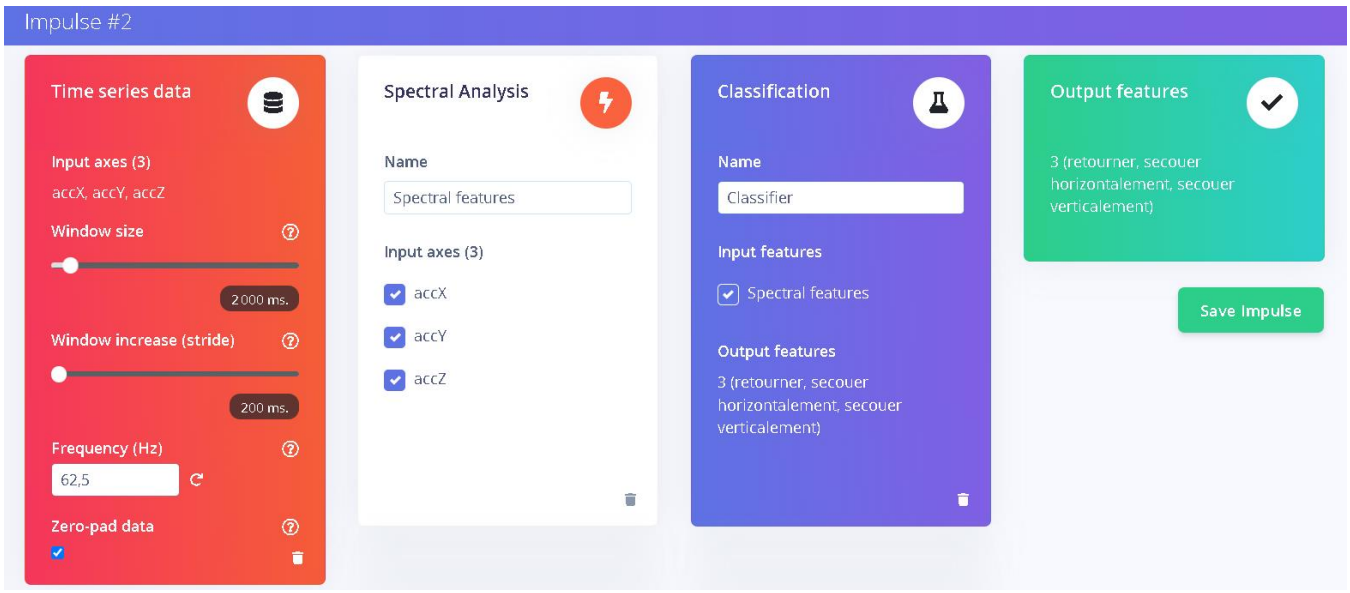
Principe de l'analyse spectrale : elle consiste à décomposer un signal en ses différentes fréquences constitutives, permettant d'étudier leur amplitude et leur répartition. Cela est habituellement réalisé à l'aide de la transformation de Fourier pour analyser des phénomènes périodiques ou vibratoires.

- Dans la fenêtre précédente (Create impulse), cliquer sur "Add a learning block" et choisir un algorithme de classification basé sur les attributs générés par l'analyse spectrale précédente.



Note : l'étape "Learning block" devrait plutôt faire partie du paragraphe 1.1.3 Choix d'un modèle, mais l'accès aux paramètres de l'analyse spectrale n'est pas accessible tant que "l'impulse" n'est pas totalement créé.

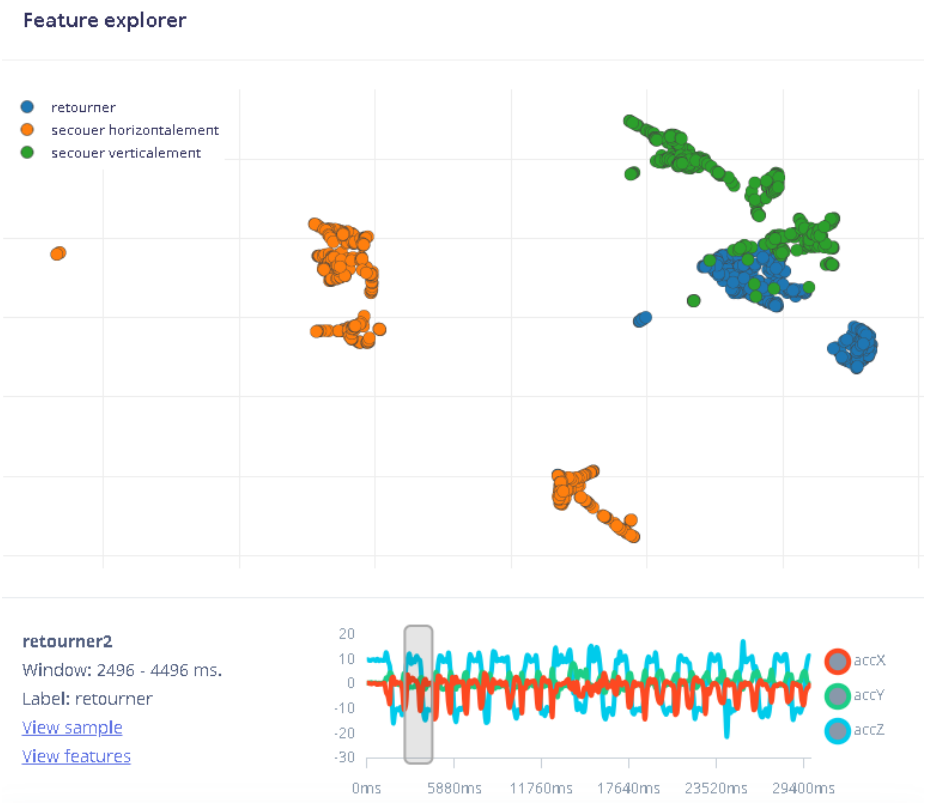
On obtient alors la fenêtre d'analyse suivante :



- Cliquer sur "Save impulse" et aller dans le menu de gauche dans "Spectral features". On peut ici configurer un filtre agissant sur les mesures avant analyse, puis choisir les paramètres de l'analyse spectrale. Dans un premier temps, on pourra conserver les paramètres par défaut.

Q3. *Parvenez-vous à distinguer l'appartenance d'un échantillon à une classe à partir de l'observation du spectre ("spectral power (log)") issu de l'analyse spectrale ? Noter les différences principales entre les spectres associés aux différences classes.*

- En haut de la fenêtre, cliquer sur "Generate features" puis lancer la génération. On génère ainsi la génération des attributs qui servirons à entrainer l'algorithme d'IA. Attendre la fin du calcul.
- On observe alors sur un graphique des ensembles de points de 3 couleurs différentes, associées aux 3 classes à classifier. Normalement, les points d'une même couleur forment des ensembles groupés (clusters). On notera que la représentation proposée en 2D fait appel à un algorithme de réduction de dimension (on a plus que 2 attributs pour chaque échantillon).



Feature explorer : exemple

Représentation (réduite à 2 dimensions) des attributs des échantillons. Ces attributs sont censés révéler des différences caractéristiques entre les 3 classes.

Dans l'exemple donné ici, on voit déjà que l'IA aura du mal à distinguer les classes bleue et verte car leurs attributs sont assez similaires. Attention néanmoins car cette proximité peut résulter de la réduction de dimension nécessaire au tracé.

Pour la classe orange, il faut dans le cas présent vérifier les mouvements effectués lors de l'acquisition car on repère 2 – 3 groupements dans la même classe.

1.1.3 Choix d'un modele

Rappel : On a déjà choisi un algorithme de classification dans l'étape 1.1.2.

- Dans le menu de gauche, aller dans "Classifier". On note qu'on nous propose d'entrainer un réseau de neurones pour effectuer la classification. Il est ici possible de modifier les caractéristiques du réseau de neurones, mais laissons dans un premier temps les paramètres par défaut.

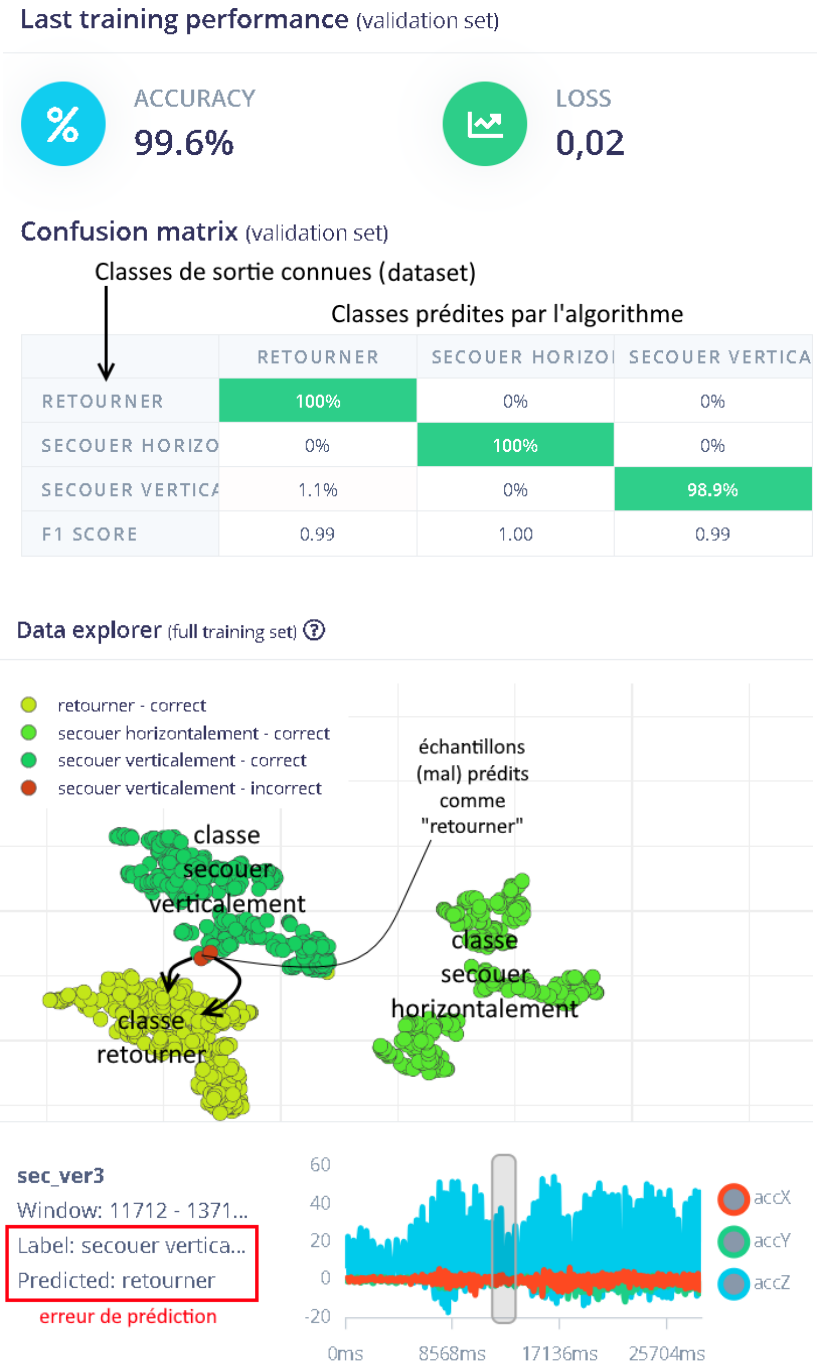
Q4. *Repérer la structure du réseau de neurones utilisé. Noter en particulier :*

- *Le nombre de neurones de la couche d'entrée*
- *Le nombre de couches cachées, et le nombre de neurones dans chaque couche cachée*
- *Le nombre de classes (donc de neurones) renvoyées par la couche de sortie.*

1.1.4 Apprentissage

- Cliquer sur "Save and train" au bas de la fenêtre. Le réseau de neurones est alors entrainé avec les données d'entrainement (75 à 80% du dataset initial). Concrètement, c'est à cette étape que Edge Impulse utilise entre autres un algorithme de descente de gradient pour ajuster les poids et les biais associés à chaque neurone artificiel (voir cours diapositives 23 et 31).

Après l'entrainement du modèle sur les données d'entrainement, Edge Impulse nous donne des statistiques sur les performances du modèle sur les données d'entrainement. On obtient entre autres une **matrice de confusion**, expliquée ci-dessous, ainsi qu'un graphique nous permettant de retrouver au travers des attributs des échantillons pour lesquels la prédiction de classe est fausse.



Analyse des performances : exemple

Le modèle a ici une précision (accuracy) de 99,6%. Cela signifie que 99,6% des échantillons des données d'entrainement ont vu leur classe de sortie prédite correctement.

Pour aller plus dans le détail, on peut observer la matrice de confusion :

- En colonnes, les classes prédites par l'algorithme.
- En lignes, les classes connues car étiquetées dans les données d'entrainement.

Si l'algorithme est précis à 100%, la diagonale de la matrice est remplie de 100%, et on a des 0 partout ailleurs.

Ici, on remarque 1,1% d'erreur sur des échantillons "secouer verticalement" qui ont été prédits avec la classe "retourner". On peut lier cette erreur à la proximité de ces classes (vert et bleu) dans le feature explorer de la page précédente.

Le "Data explorer" nous permet de retrouver les échantillons pour lesquels la prédiction a été mauvaise. Ci-contre, on peut voir que 2 échantillons ont vu leur classe mal prédite.

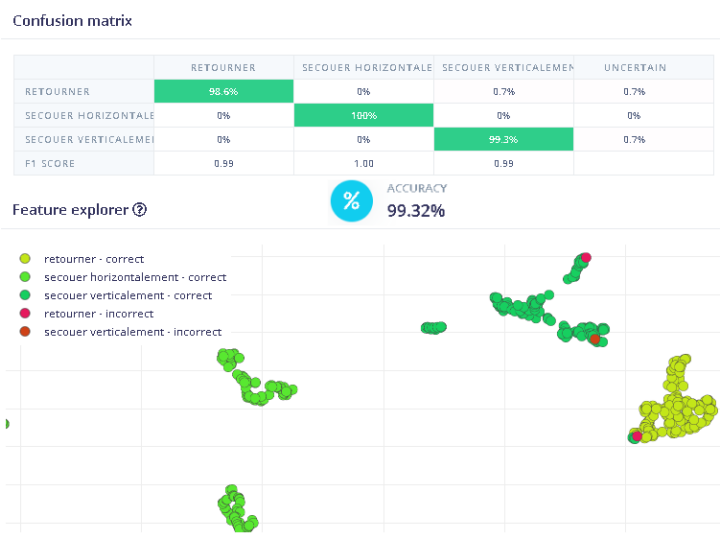
1.1.5 TEST DU MODELE

Edge Impulse a déjà testé le modèle sur l'ensemble des données d'entrainement (75 à 80% du dataset initial). Il est également nécessaire de tester l'algorithme sur des données "qu'il n'a jamais vues" mais dont on connaît le résultat (la classe) attendue pour évaluer ses performances. C'est pourquoi on effectue un test similaire au précédent sur l'ensemble de données de test (20 à 25% du dataset initial).

- Dans le menu de gauche, cliquer sur "Model testing" et lancer la classification sur les données de test en cliquant sur "classify all".

On obtient un récapitulatif de performances similaire au précédent. Assez logiquement, les performances sont un peu moins satisfaisantes que sur le set de données d'entrainement (on rappelle que l'algorithme n'avais "jamais vu" les données de test).

Dans l'exemple ci-contre, la précision obtenue, supérieure à 99% est satisfaisante.



Q5. *Dans votre cas, observer les matrices de confusion obtenue sur les données d'entrainement et sur les données de test. Quelles différences relevez-vous ? Parvenez-vous à expliquer pourquoi certains échantillons ont été mal prédits ?*

1.1.6 AJUSTEMENT DES HYPERPARAMETRES

- Si la précision de prédiction n'est pas satisfaisante, il faut dans l'ordre :

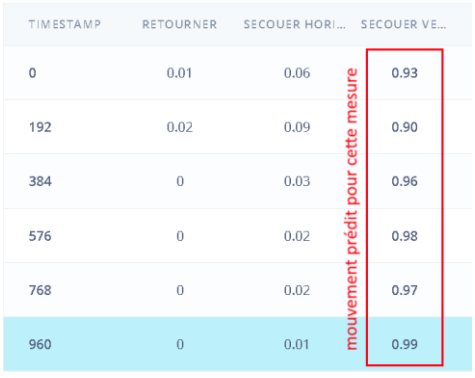
1. Eventuellement remettre en question la pertinence de certaines données "mal" acquises. Dans le menu "Data acquisition", on peut désactiver, tronquer, ou séparer en plusieurs morceaux certaines mesures si elles ne sont pas pertinentes.
2. Modifier les paramètres du bloc "Spectral features", principalement en modifiant le filtrage des données.
3. Modifier les paramètres du bloc "Classifier", en modifiant les paramètres du réseau de neurones.

1.1.7 INFERENCE ➔ PREDICTIONS

On entre ici dans la phase d'utilisation (inférence) de l'algorithme d'IA qui a été précédemment entrainé et testé.

<u>Première possibilité avec Edge Impulse :</u> "Live Classification" – adapté au test rapide du modèle

- Aller dans le menu de gauche dans "Live classification". Reconnecter le téléphone en actualisant la page ou en reflashant le QR code s'il s'est déconnecté.
- Choisir une durée d'acquisition de 3s = 3000ms par exemple, cliquer sur "Start sampling" et effectuer un des trois mouvements à distinguer.
- A la fin de l'acquisition, la mesure apparaît à l'écran de l'ordinateur. On retrouve le découpage en plusieurs échantillons avec la méthode de la fenêtre glissante définie en 1.1.2, ce qui amène plusieurs échantillons pour une seule mesure. Pour chaque échantillon, l'algorithme effectue une prédiction d'appartenance à une des trois classes. Cette prédiction est caractérisée par une probabilité (entre 0 et 1) d'appartenance à chaque classe.



<u>Seconde possibilité avec Edge Impulse :</u> Déploiement pour utilisation

On peut ici déployer le modèle d'IA créé pour une utilisation sur un appareil. Cet appareil peut être :

- Un smartphone ➔ Utilisation sous forme d'une WebApp (local - pas d'accès réseau nécessaire)
- Un microcontrôleur type Arduino ou autre ➔ Bibliothèque logicielle compilée (bibliothèque C++, Arduino…). Le fonctionnement de la bibliothèque ne nécessite pas non plus d'accès réseau.

- Aller dans le menu de gauche dans "Deployment". Scanner le QR Code situé sur la droite de l'écran avec votre smartphone.
- Attendre la compilation du projet sous la forme d'une WebApp. A la fin de la compilation, l'inférence démarre automatiquement : acquisition du signal puis classification comme précédemment. La WebApp peut être téléchargée et conservée en local sur le smartphone. Son utilisation ne nécessite aucun accès réseau.

Bravo, vous avez entrainé, testé et déployé votre première IA de classification !

Q6. *Tester le fonctionnement du modèle pour les différents mouvements. Que se passe-t-il si l'on effectue un mouvement différent de ceux des trois classes prévues ? Que se passe-t-il si on laisse le smartphone statique ? Proposer une solution à ce problème.*

1.1.8 POUR ALLER PLUS LOIN : DETECTION D'ANOMALIE

- Ajouter au modèle précédent une classe "statique" associée au smartphone immobile. Refaire des mesures pour différentes situations considérées comme statiques : posé sur une table, en main sans bouger… Mettre à jour le modèle en balayant toutes les étapes pour avoir finalement un modèle effectuant une prédiction parmi 4 classes.

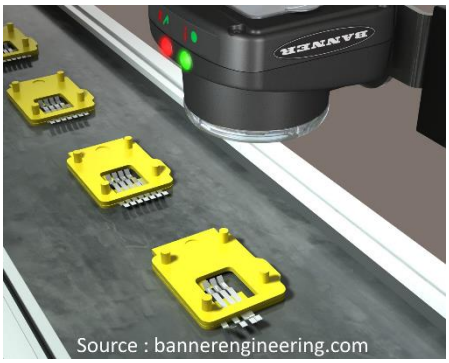
<u>Détection d'anomalie : principe et intérêt</u>

Cette détection permet de distinguer un mouvement inconnu, c'est-à-dire qui n'appartiendrait à aucune des 4 classes précédemment définies. Un tel système peut avoir un réel intérêt pour un service maintenance dans une entreprise :

- On identifie sur une machine différents modes de fonctionnement dits "normaux". Chacun des modes est défini comme une classe et un algorithme similaire au précédent nous permet de dire dans quel mode fonctionne la machine à un instant donné.
- Une détection d'anomalie détecte un mode de fonctionnement anormal, qui nécessite l'intervention d'un service maintenance. Celui peut alors être averti avant un nécessaire arrêt de ligne.
- Dans le futur, une anomalie (panne) récurrente nécessitant un changement de composant peut être défini comme une classe connue. On peut grâce à l'algorithme d'IA anticiper les changements de composants.

- Dans le menu de gauche, retourner dans "Create impulse". Ajouter un second "learning block" correspondant à la détection d'anomalie ("Anomaly Detection (K-means) "). On voit ici que cette détection utilise un algorithme des k-moyens (voir cours diapositive 27). Cliquer sur "Save impulse".
- Dans le menu de gauche, aller dans "Anomaly detection". On peut ici sélectionner les différents attributs à prendre en compte pour la détection d'anomalies, parmi tous les attributs (features) définis à l'issue de l'étape 1.1.2. Edge Impulse propose d'en sélectionner certains, correspondant habituellement aux accélérations efficaces sur les différents axes (voir rappel d'une valeur efficace d'un signal périodique). Sélectionner les attributs proposés par défaut puis cliquer sur "Save and train".
- Tester le fonctionnement de la détection d'anomalie en utilisant une des deux méthodes du paragraphe 1.1.7.

1.2 DETECTION D'ELEMENTS SUR IMAGES



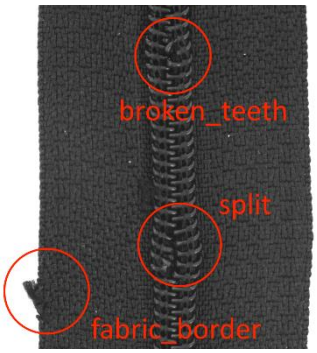
Dans cette partie, Edge Impulse sera utilisé pour distinguer des défauts pré-identifiés (connus) présents sur des images de pièces issues d'une ligne de fabrication (par exemple). On fait ici appel à un réseau de neurones de convolution (CNN) – voir cours diapositive 32. Nous allons ici entrainer un réseau de neurones avec un nombre d'images relativement faible. Pour permettre un apprentissage pertinent, Edge Impulse utilise le bloc "Transfer learning" qui part d'un réseau de neurones (FOMO) entrainé sur une base de données généraliste (MobileNetV2) à reconnaitre de multiples objets et en réentraine seulement une partie pour le cas spécifique des images qu'on lui communique.

Dans une entreprise, on peut imaginer repérer sur des images des différents défauts connus (on ne parle pas ici de défauts inconnus). Dans le cas présent, on propose d'utiliser une base de données existante présentant des images issues de cas industriels : le dataset MVTec Anomaly Detection présente plusieurs exemples de pièces avec des défauts identifiés. Cette base de données est normalement destinée à un apprentissage non supervisé, mais nous allons l'utiliser dans le cas d'un apprentissage supervisé, c'est-à-dire qu'il faudra étiqueter manuellement les défauts sur les images du dataset.

Si jamais vous souhaitez utiliser une reconnaissance d'objets chez vous ou en entreprise, attention à utiliser des conditions d'éclairement et de fond qui seront celles lors de l'inférence. Si l'apprentissage est effectué sur des photos bien éclairées et sans ombre, l'algorithme rencontrera des difficultés à reconnaitre les objets dans d'autres situations moins favorables.

L'ensemble des consignes de cette partie sera volontairement moins guidé puisque vous avez découvert le fonctionnement de la plateforme dans le paragraphe 1.1.

1.2.1 COLLECTE DES DONNEES



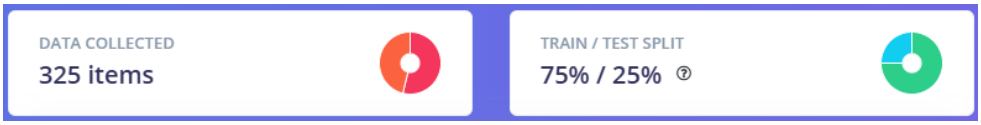
On aurait ici pu acquérir des images à partir de l'appareil photo d'un smartphone ou à partir d'une caméra, et effectuer des prises de vues de pièces avec différents défauts / conformes. Afin de gagner du temps, on utilise le dataset Zipper de MVTec AD qui a été légèrement remanié pour cet exemple : on cherche à détecter et identifier des défauts sur des fermetures éclair (textiles) sortant d'une ligne de fabrication.

- Sur Edge Impulse, créer un nouveau projet nommé "Defects detection".
- Explorer le dossier "Zipper dataset" proposé : identifier les images d'entrainement et celles de test de l'algorithme. Parmi les images d'entrainement, repérer les différents défauts connus. Choisir dans un premier temps 2 défauts que vous souhaitez détecter, par exemple "split" et "broken_teeth".
- Dans l'onglet Data acquisition, cliquer sur "Upload data" et uploader **dans le dataset d'entrainement (Training)** toutes les images contenant le 1er défaut connu à détecter ("split" par exemple).

Nous arrivons maintenant dans la phase compliquée lorsqu'on travaille avec des images : l'étiquetage (labeling en anglais). Il s'agit d'associer une classe à une région de l'image, et non pas à la globalité de l'image. En effet, il se peut qu'il y ait plusieurs éléments à détecter (défauts connus ici) sur une même image.

- Sur le haut de la fenêtre, cliquer sur "Labeling queue". On arrive à une fenêtre dans laquelle on peut esquisser des rectangles (bounding box) autour du défaut, puis l'étiqueter - ici avec le label "split". S'il y a plusieurs défauts sur la même image, on peut esquisser plusieurs bounding boxes.
- Effectuer l'étiquetage de toutes les images contenant le défaut "split".

- Importer les images associées au second défaut connu choisi ("broken_teeth" par exemple) **dans le dataset d'entrainement**, puis effectuer l'étiquetage comme précédemment.
- Importer les images issues contenues dans le dossier test/combined_defects **dans le dataset de test**. On se sert ici de ces données pour tester le modèle. Effectuer l'étiquetage comme précédemment. Chacune de ces images présente plusieurs défauts à étiqueter.
- Importer les images issues contenues dans le dossier train/good **dans le dataset d'entrainement**. Il n'est pas nécessaire d'étiqueter ces données puisqu'elles ne présentent aucun défaut à identifier.
- Importer les images issues contenues dans le dossier test/good **dans le dataset de test**. Il n'est pas nécessaire d'étiqueter ces données puisqu'elles ne présentent aucun défaut à identifier.
- Vérifier l'existence de 2 classes parmi les données étiquetées, et que la répartition set d'entrainement / de test est pertinente :



Remarque sur l'étiquetage des données :

Nous venons de voir que l'étiquetage des données est une phase longue et pénible. Il devient peu à peu possible d'automatiser cet étiquetage, nécessaire dans le cas d'un apprentissage supervisé, en utilisant une IA et un LLM auquel on communique des règles (nouveauté Edge Impulse 2024).

On a ici des possibilités avancées. On peut par exemple :

- Repérer un objet donné sur une image puis l'étiqueter en fonction de sa couleur
- Repérer un objet donné sur une image puis en extraire une caractéristique numérique : taille, niveau de remplissage… (utilisation d'une API OpenAI par exemple – réservé aux versions payantes de ChatGPT)

Dans le cas présent, en se limitant aux possibilités offertes par la version gratuite de la plateforme, **l'étiquetage automatique fonctionne mal pour les exemples de la base MVTec** puisqu'il faut arriver à caractériser proprement les défauts dans un prompt (càd par une description textuelle).

Si vous souhaitez essayer avec d'autres images à la fin du TP, par exemple des images de crayon, voici la démarche

- Dans le menu du haut de l'écran, aller dans "AI labeling". Sélectionner dans Step 1 : Bounding box labeling with OWL-ViT. Ici on va uniquement lui demander de reconnaitre les Crayons. Pour cela entrer comme prompt : A pencil (Crayon,0.1). Ceci signifie que tout objet identifié comme "A pencil" avec une probabilité d'au moins 10% se verra attribuer la classe Crayon.
- Tester le fonctionnement de l'étiquetage automatique en cliquant à droite sur "Label preview data" et vérifier que les crayons sont bien étiquetés par le label Crayon. Si ce n'est pas satisfaisant, modifier les paramètres.
- Une fois le test concluant, cliquer sur "Label all data" et attendre la fin de l'étiquetage automatique.
- Vérifier dans le dataset que les images ont été correctement étiquetées. Corriger manuellement si nécessaire.

1.2.2 MISE EN FORME DES DONNEES

Le CNN FOMO est ici préentraîné sur une version de la base de données MobileNetV2 comportant des images carrées de 96x96 pixels. On réduit donc toutes les images à ce format avant d'en extraire les attributs.

- Créer un "impulse" comme dans l'étape 1.1.2. Y entrer une réduction de dimension 96x96 et un "resize mode" de type "squash" afin de ne pas perdre une zone de l'image initialement rectangulaire. Ajouter

ensuite un processing block "Image" et un learning block "Object detection". Terminer en cliquant sur "Save impulse".

- Dans le menu "Image" à gauche, vous avez le choix de conserver l'image en couleur (RGB) ou de faire le traitement en niveaux de gris qui correspond à un problème 3 fois plus simple à résoudre. Dans notre cas, on peut raisonner sur des images en niveau de gris.
- Terminer en générant les attributs qui vont alimenter le réseau de neurones : cliquer sur "Generate features".

1.2.3 CHOIX D'UN MODELE

- Dans le menu "Object detection" à gauche, on retrouve les paramètres d'apprentissage que l'on peut ajuster. On laisse ici les paramètres par défaut mais on peut activer la fonction "Data augmentation" : elle augmente le nombre de données du dataset en effectuant des opérations diverses sur les images existantes (rotations, mises à l'échelle, introduction de bruit, modification du contraste ou de la luminosité…).

1.2.4 APPRENTISSAGE

- Dans le même menu, cliquer sur "Save and train" et attendre.

Remarque : le temps de calcul est dépendant de la complexité du problème. On note qu'il est ici beaucoup plus important que dans l'exemple n°1 car le traitement d'images est gourmand en ressources. Attention également à la consommation énergétique qui est également proportionnelle à ce temps de calcul.

- Observer la matrice de confusion obtenue suite à l'entrainement du modèle.

Q7. *Avez-vous obtenu des résultats satisfaisants avec cet algorithme ? Identifier les points bloquants dans le développement de ce modèle d'IA.*

1.2.5 TEST DU MODELE

- Tester le modèle sur les données de test comme effectué au paragraphe 1.1.5.

Q8. *L'identification des défauts connus est-elle concluante sur les données de test ? Comment améliorer la détection de défauts connus proposée ?*

1.2.6 AMELIORATIONS

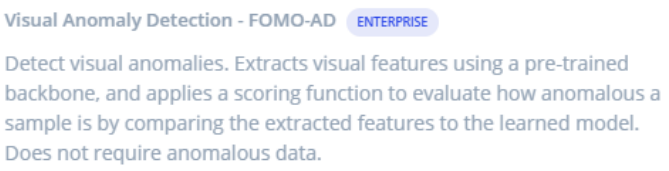
Si le temps le permet, effectuer des améliorations sur le modèle d'IA précédemment créé.

1.2.7 CAS DES DEFAUTS INCONNUS

Nous avons traité ici le cas de défauts connus et déjà identifiés.

Il est également possible d'implémenter des algorithmes visant à détecter des défauts inconnus : on parle ici d'apprentissage non supervisé puisque les défauts inconnus ne sont pas étiquetés.

Pour implémenter une telle détection, Edge Impulse propose un algorithme FOMO Anomaly Detection disponible parmi les "learning blocks" mais réservée aux clients Entreprise (version payante). Elle utilise des images correspondant uniquement à des pièces conformes et recherche des éléments qui en diffèrent.



On notera qu'un défaut détecté plusieurs fois par un tel algorithme peut ensuite être étiqueté comme un défaut connu et être intégré à l'algorithme mis au point précédemment.

PARTIE 2 : DEPLOIEMENT SUR UN MICROCONTROLEUR

Les exemples de cette partie sont basés sur le travail publié par Seeed Studio sur leur site.

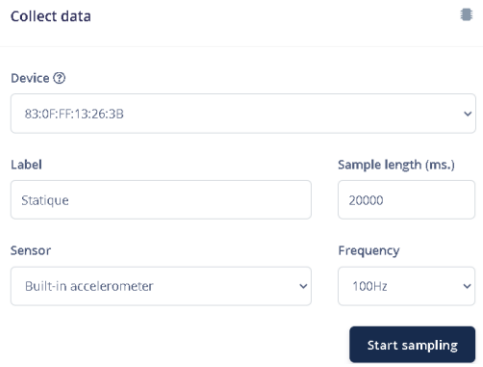
Nous allons dans cette partie utiliser Edge Impulse pour implémenter un algorithme d'IA dans un microcontrôleur, ici le Wio terminal utilisé lors du TP 1.

2.2 MISE EN PLACE DE LA LIAISON WIO TERMINAL – EDGE IMPULSE

Pour envoyer les données à la plateforme Edge Impulse, nous allons utiliser une connexion du type WebSerial via le navigateur Edge ou Chrome (les autres ne supportent pas cette fonction pour le moment). Le navigateur transfère alors les données arrivant sur le port série (USB) vers internet. Dans Edge Impulse, ce type de connexion est nommé WebUSB.

- Connecter le Wio terminal à l'ordinateur, puis glisser 2 fois le bouton de redémarrage (côté gauche) vers le bas. On entre dans le mode bootloader, et un dossier nommé Arduino s'ouvre.
- Coller dans ce dossier le fichier "edge_impulse_firmware.uf2" disponible dans le dossier du TP ou au lien suivant. A la fin de l'upload, la fenêtre se ferme automatiquement.
- Aller sur la plateforme Edge Impulse. Créer un nouveau projet nommé. Aller dans le menu de gauche "Data acquisition" et cliquer sur l'icone symbolisant la connexion USB. Dans le menu qui s'ouvre, sélectionner le port COM associé au Wio terminal.

- Paramétrer le menu d'acquisition comme sur la figure ci-contre : acquisition avec l'accéléromètre intégré au Wio terminal, durée des échantillons 20s, fréquence d'échantillonnage 100Hz.
- Cliquer sur Start sampling pour vérifier le bon fonctionnement de l'acquisition.



Remarque 1 : Si vous ne parvenez pas à vous connecter depuis Edge Impulse, refaire l'upload du firmware Edge Impulse vers la carte (ci-dessus).

Remarque 1 : On peut bien sûr ici utiliser un capteur différent (parmi ceux proposés) si l'on souhaite obtenir des mesures d'une autre grandeur physique.

2.3 ELABORATION D'UN ALGORITHME D'IA – EXEMPLE – RECONNAISSANCE DE MOUVEMENT

On cherche ici à reproduire avec le Wio Terminal l'activité effectuée avec le smartphone au paragraphe 1.1. On pourrait se contenter d'implémenter dans le Wio terminal l'algorithme développé pour le smartphone, mais les accéléromètres sont des modèles différents (bandes passantes et réponses différentes).

En général, il est préférable d'acquérir le dataset avec le capteur qui sera utilisé durant l'inférence.

2.3.1 COLLECTE ET TRAITEMENT DES DONNEES – CREATION ET TEST DU MODELE

- Collecter environ 1 minute de données pour les trois classes suivantes :

  - Statique : pas de mouvement
  - Mouvement 1 : à définir
  - Mouvement 2 : à définir

- Effectuer la séparation du dataset ➔ Données d'entrainement / données de test.
- Créer un impulse adapté comme vu précédemment.

- Jouer cette fois sur les paramètres du filtre dans "Spectral features" : on implémente un filtre passe bas qu'on règle pour conserver la composante principale du mouvement à reconnaître.
- Créer puis entrainer le modèle d'IA. Modifier les réglages si les performances ne sont pas satisfaisantes.
- Tester le modèle sur les données de test.

2.3.2 *Deploiement du modele sur un microcontroleur*

- Depuis Edge Impulse, aller dans le menu de gauche dans "Delployment" et sélectionner "Arduino library".
- Cliquer sur "Build", puis attendre la génération et le téléchargement de la bibliothèque.
- Démarrer le logiciel Arduino IDE, aller dans le menu Croquis / Inclure une bibliothèque / Ajouter une bibliothèque .ZIP. Sélectionner la bibliothèque qui vient d'être téléchargée et l'installer.
- Ouvrir le fichier "Wio_reconnaissance_mvt.ino" donné dans le dossier du TP. Ce fichier a déjà été adapté à la reconnaissance de 3 classes différentes à partir de signaux acquis via l'accéléromètre du Wio terminal.
- Modifier à la ligne 23 le nom du modèle d'IA à utiliser, associé à la bibliothèque que vous venez d'installer :

```
24  #include <Wio_mvt_inferencing.h> // Bibliothèque Edge Impulse : "nom_projet"_inferencing.h
```

- Adapter les lignes 159 à 181 de manière à afficher correctement le nom de vos classes (l'ordre correspond à celui utilisé dans la matrice de confusion sur la plateforme Edge Impulse) :

```
159 if (result.classification[0].value > 0.5) {
160  tft.fillScreen(TFT_BLUE);
161  tft.setFreeFont(&FreeSansBoldOblique12pt7b);
162  tft.drawString("Cercle horizontal", 20, 80);
163  delay(1000);
164  tft.fillScreen(TFT_WHITE);
165 }
166
167 if (result.classification[1].value > 0.5) {
168  tft.fillScreen(TFT_RED);
169  tft.setFreeFont(&FreeSansBoldOblique12pt7b);
170  tft.drawString("Secouer verticalement", 20, 80);
171  delay(1000);
172  tft.fillScreen(TFT_WHITE);
173 }
174
175 if (result.classification[2].value > 0.5) {
176  tft.fillScreen(TFT_GREEN);
177  tft.setFreeFont(&FreeSansBoldOblique12pt7b);
178  tft.drawString("Statique", 20, 80);
179  delay(1000);
180  tft.fillScreen(TFT_WHITE);
181 }
```

→ Classe "Cercle horizontal" affichée sur un écran bleu si probabilité classe > 0,5

→ Classe "Secouer vertical" affichée sur un écran rouge si probabilité classe > 0,5

→ Classe "Statique" affichée sur un écran vert si probabilité classe > 0,5

- Installer si nécessaire les bibliothèques manquantes (LIS3DHTR par exemple) en effectuant une recherche depuis le gestionnaire de bibliothèques.
- Compiler puis téléverser le programme vers le Wio Terminal.
- Observer le résultat – effectuer le mouvement pour vérifier que les classes sont correctement prédites.

Q9. *Les classes sont-elles correctement prédites ? Peut-on améliorer la classification et comment ?*

2.4 POUR ALLER PLUS LOIN : PROJET

Les exemples proposés dans cette partie sont basés sur les travaux publiés par Seeed Studio sur leur site.

A votre tour d'imaginer un algorithme d'IA embarquée dans le Wio Terminal, alimenté par des données de capteurs. On pourra prendre comme base l'un des projets proposés sur le site de Seeed Studio, pour en proposer une utilisation dans un contexte industriel :

- Classification à partir des vibrations (base similaire à l'exemple traité en 2.3) : Identification des modes de marche standard d'une machine + ajout d'une détection d'anomalie pour la maintenance prédictive.
- Reconnaissance de son : Utilisation de cet exemple pour effectuer une classification des modes de fonctionnement d'un système émettant du son (pompe, machine-outil…).
- Comptage avec un capteur à ultrasons : Comptage des véhicules entrant / sortant d'une zone dans une entreprise.
- Classification de substances : Test de la conformité à la norme de différents panneaux de particules de bois, concernant l'émission de formaldéhydes dans l'air.

Illustration générée avec ChatGPT