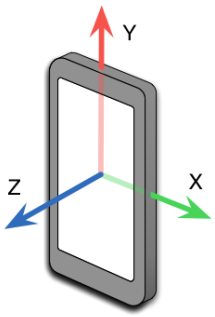
TP 2 – TinyML – IA embarquee

Elements de corrige

Partie 1 : Deploiement sur un smartphone

1.1 Classification de mouvement

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



Exemples d'utilisation d'une classification de mouvement sur un smartphone :

- Détection de chute d'une personne / d'accident de transport.
- Classification et mesure de durée associée à plusieurs modes de déplacement : marche / course / vélo / voiture / train…
- Commande d'une interface (objet externe ou application interne au smartphone) par un geste.

Définition de 3 mouvements (qui seront 3 classes) à distinguer.

- Secouer l'appareil verticalement (perpendiculairement à l'écran)
- Secouer l'appareil horizontalement (dans le plan de l'écran – 2 directions !)
- Retourner l'appareil (différents axes)

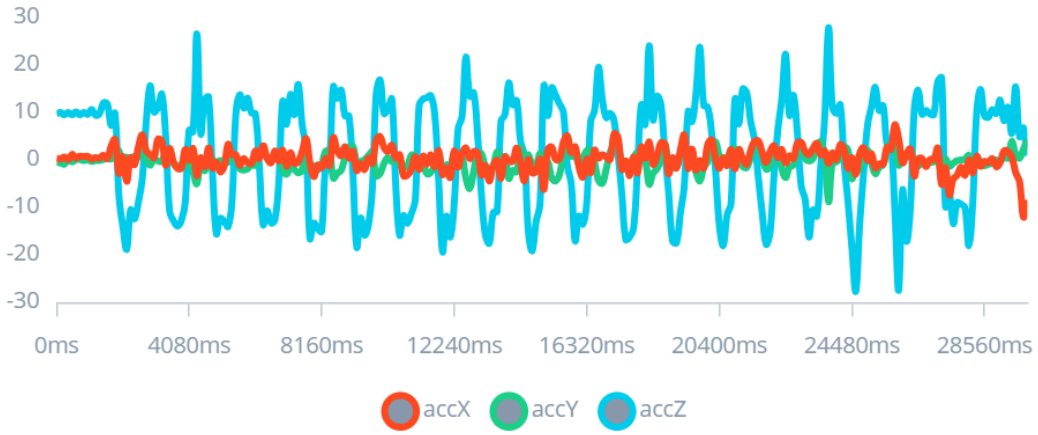
1.1.1 Collecte des donnees

En suivant les consignes du sujet, on parvient à enregistrer au moins 2 minutes de mesures pour chacun des 3 mouvements définis précédemment. On insiste sur la diversité parmi les classes : les acquisitions doivent faire apparaître tout type de mouvement devant être reconnu dans chaque classe.

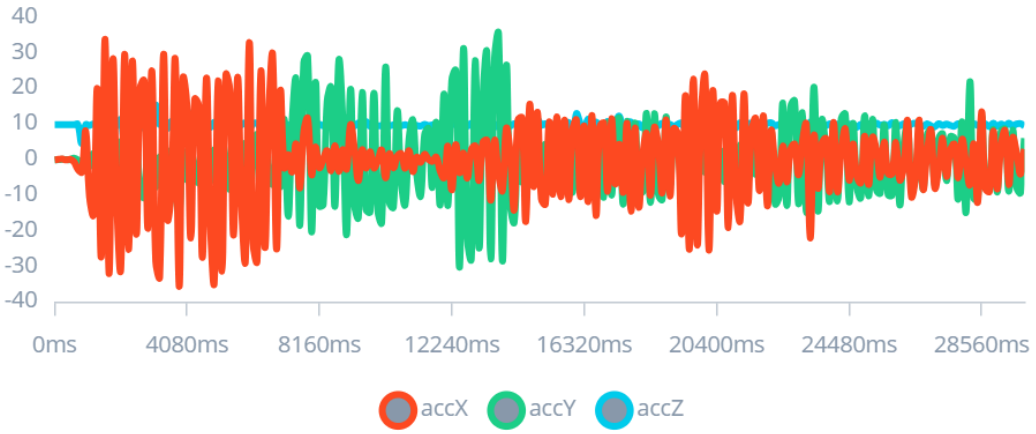
Q2. *Observer les allures des courbes d'accélération obtenues pour chacune des trois classes. Notez-vous des caractéristiques distinguant chacune des classes ? Observez-vous également de la diversité parmi les classes (variations de l'allure des signaux au sein d'une même classe) ?*

Observons pour commencer les courbes d'accélération obtenues pour un échantillon appartenant à chacune des trois classes de mouvements à identifier. On peut noter des tendances dans ces échantillons.

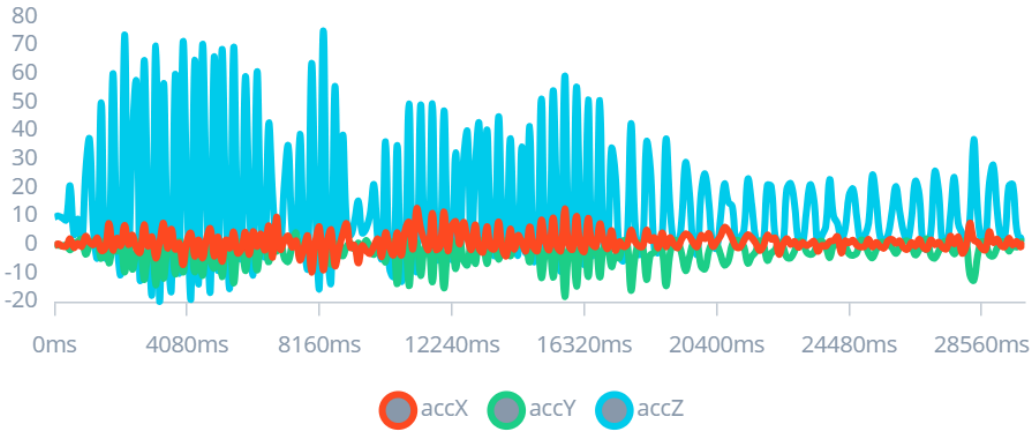
Exemple d'échantillon de la classe retourner ➔ $accX \approx accY \approx 0$, $accZ$ oscillant entre 2 valeurs symétriques (environ 10m/s² car accélération gravitationnelle qui s'inverse dans le repère du smartphone lors du retournement).



Exemple d'échantillon de la classe secouer horizontalement ➔ $accZ \approx 10\ m/s^2$ (accélération gravitationnelle), $accX$ et $accY$ oscillant autour de 0 de manière symétrique (soit l'une, soit l'autre, soit les deux simultanément). On distingue ici la diversité parmi cette classe (non mise en évidence pour les autres classes mais également présente).



Exemple d'échantillon de la classe secouer verticalement ➔ $accX \approx accY \approx 0$, $accZ$ oscillant autour de 10 m/s².



On perçoit ici des différences entre les échantillons associés aux différentes classes. Repérer ces différences via des lignes de code implémentant du filtrage et de la transformée de Fourier serait envisageable mais complexe. C'est pourquoi nous allons dans la suite entrainer un algorithme comportant un réseau de neurones pour reconnaitre ces mouvements. Une couche de prétraitement placée en amont permettra le calcul du spectre (transformée de Fourier) associé aux échantillons, qui sera donné en entrée du réseau de neurones.

On sépare ensuite l'ensemble des échantillons obtenus en un dataset d'entrainement et un dataset de test selon un ratio de données de test habituellement compris entre 1/4 et 1/3 du dataset initial.

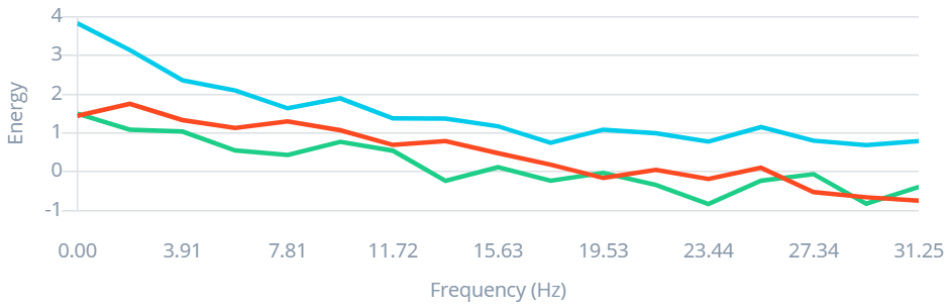
1.1.2 Mise en forme des donnees

Le prétraitement des données (mise en forme) est effectué comme proposé dans le sujet. On note que la durée nommée « Window size » doit être adaptée à la nature des mouvements : elle doit englober au moins une période dans le cas d'un mouvement périodique (oscillation).

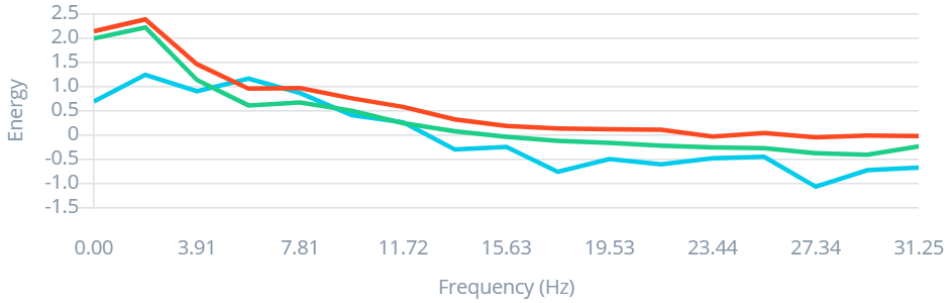
Q3. *Parvenez-vous à distinguer l'appartenance d'un échantillon à une classe à partir de l'observation du spectre ("spectral power (log)") issu de l'analyse spectrale ? Noter les différences principales entre les spectres associés aux différences classes.*

Les spectres associés à un échantillon (de durée 2s) appartenant à chaque classe sont donnés sur les figures ci-dessous.

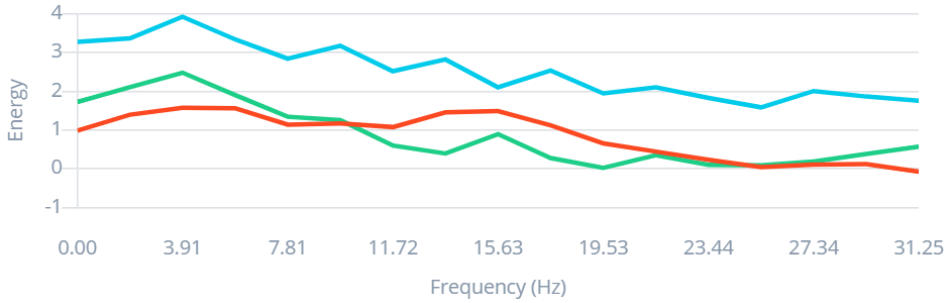
Exemple d'un échantillon de la classe retourner :



Exemple d'un échantillon de la classe secouer horizontalement :



Exemple d'un échantillon de la classe secouer verticalement :



On note des différences entre les spectres. Par exemple la classe secouer horizontalement se traduit par des densités spectrales importantes sur les accélérations suivant X et Y (respectivement rouge et vert), tandis que les classes secouer verticalement et retourner présentent des densités spectrales importantes sur l'accélération suivant Z (bleu). Ceci est normal puisque le mouvement secouer horizontalement ne présente en théorie aucun mouvement selon Z (axe perpendiculaire à l'écran du smartphone). La différence entre les classes secouer verticalement et retourner est plus difficile à détecter sur les spectres, mais on repère des allures de spectres différentes.

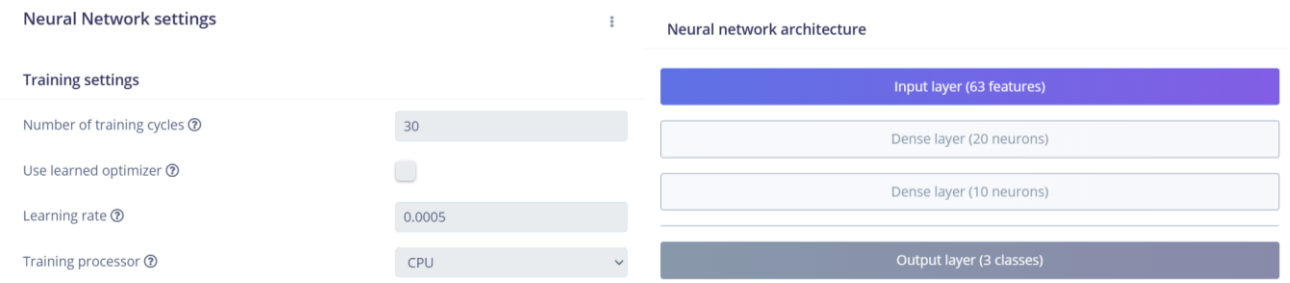
L'intérêt de l'utilisation d'un réseau de neurones se trouve ici : en lui communiquant suffisamment de données d'entrée (d'apprentissage), le réseau de neurones pourra déceler les différences entre les classes à partir de données d'entrée complexes, et prédire une probabilité d'appartenance à une classe de sortie.

Après avoir généré les Features, on observe 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). Comme évoqué dans le sujet du TP, les classes retourner (bleu sur la figure ci-dessous) et secouer verticalement (vert sur la figure ci-dessous) sont proches, voire mélangées ce qui montre qu'elles seront plus délicates à distinguer. Cette conclusion a déjà été formulée à partir de l'observation des spectres de la page précédente.

On notera que la représentation proposée en 2D fait appel à un algorithme de réduction de dimension (on a ici 63 attributs pour chaque échantillon) dont les caractéristiques ne sont pas explicitement données. Le fait d'avoir des clusters mal séparés peut provenir de cet algorithme, car l'observation des multiples Features (63 ici) fait encore apparaître de nettes différences entre les classes. C'est le cas ici au vu des performances de l'algorithme obtenues dans la suite.

1.1.3 CHOIX D'UN MODELE

Les propriétés du réseau proposé par défaut sont les suivantes :



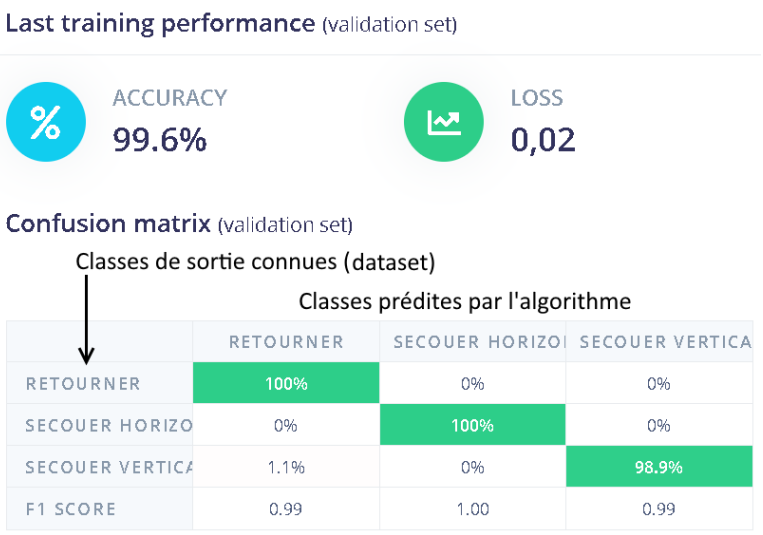
Q4. *Repérer la structure du réseau de neurones utilisé. Noter en particulier :*

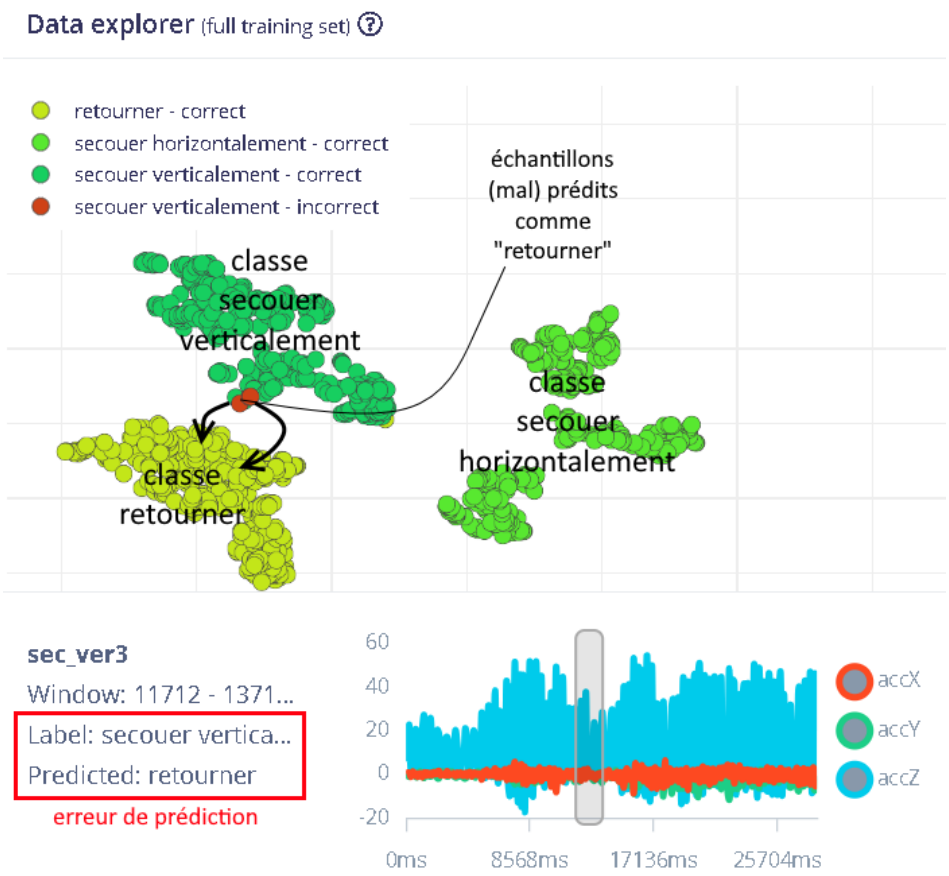
- Le nombre de neurones de la couche d'entrée : 63 neurones d'entrée – ce chiffre correspond au nombre d'attributs (Features) définis précédemment à partir de l'analyse spectrale.

- Le nombre de couches cachées, et le nombre de neurones dans chaque couche cachée : une couche cachée de 20 neurones suivie d'une seconde couche cachée de 10 neurones. Le choix de l'architecture interne du réseau n'est pas simple et ne sera pas abordé ici.
- Le nombre de neurones de sortie : 3, correspondant aux 3 classes attendues en sortie. On note que ces neurones nous renverront une probabilité d'appartenance à chaque classe.

1.1.4 APPRENTISSAGE

L'analyse des résultats et performances est présentée dans le sujet. Il est intéressant de la mettre en œuvre pour les données acquises par chacun, pour bien percevoir les origines des erreurs.





On note ici 2 échantillons appartenant en réalité à la classe secouer verticalement qui ont été prédits comme appartenant à la classe retourner. Nous avions déjà évoqué précédemment la possibilité d'erreurs de classification entre ces deux classes. Les mouvements sont peut-être trop similaires dans certaines situations.

1.1.5 TEST DU MODELE

En effectuant le test du modèle sur le dataset de test (que l'algorithme n'a pas utilisé pour son entrainement), on obtient une nouvelle matrice de confusion :

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 ?*

La précision obtenue est légèrement inférieure à celle obtenue sur les données d'entrainement, mais reste tout de même très satisfaisante. On note que les mauvaises prédictions peuvent provenir de « phases » durant lesquelles le mouvement effectué pour une classe était moins prononcé : par exemple, durant une acquisition de la classe secouer verticalement, le mouvement a été stoppé / ralenti pendant une durée proche de celle de la largeur d'une fenêtre glissante permettant la définition des échantillons.

1.1.6 AJUSTEMENT DES HYPERPARAMETRES

On parvient normalement à des performances satisfaisantes en suivant la démarche proposée. Si ce n'est pas le cas, revoir les points indiqués dans le sujet.

1.1.7 INFERENCE ➔ PREDICTIONS

On déploie le modèle sur le smartphone et on effectue la phase d'inférence, qui propose une probabilité d'appartenance à chacune des classes.

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.*

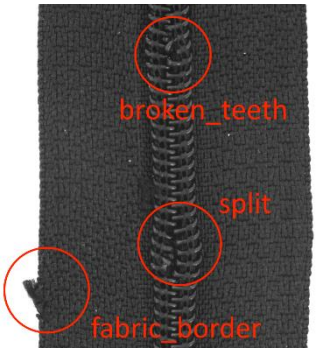
Si on effectue un mouvement n'appartenant à aucune des classes, le modèle essaie tout de même de le classifier parmi une des classes existantes. Le modèle n'a pas été créé et entrainé pour cela. Pour contourner cela, on peut implémenter une détection d'anomalie, qui signalera un échantillon reconnu comme n'appartenant à aucune des classes prédéfinies (voir partie suivante).

Si le smartphone est statique, le modèle essaie là encore de classifier le (non) mouvement dans une des classes existantes. Une solution pour contourner cela consiste à créer une nouvelle classe correspondant à l'absence de mouvements.

1.1.8 POUR ALLER PLUS LOIN : DETECTION D'ANOMALIE

L'ajout d'une nouvelle classe « statique » au modèle, et d'une détection d'anomalie le rend plus complet. Les prédictions alors effectuées deviennent plus satisfaisantes, car elles permettent de prendre en compte des mouvements autres que ceux identifiés dans les 3 classes initiales.

1.2 DETECTION D'ELEMENTS SUR IMAGES



1.2.1 COLLECTE DES DONNEES

Le suivi des consignes du sujet permet d'obtenir un dataset entrainement + test contenant des images de fermetures éclair conformes, et de fermetures éclair comprenant un défaut « split », un défaut « broken » ou plusieurs des deux défauts précédents sur la même image. L'étiquetage des données est une phase longue, fastidieuse, et nécessitant une bonne connaissance des classes (ici des défauts) à identifier.

Remarque – apprentissage supervisé :

On utilise ici un apprentissage supervisé, c'est-à-dire qu'on identifie des défauts types connus à reconnaitre sur les images. On pourrait également utiliser un apprentissage non supervisé, qui serait entrainé uniquement sur des images de fermetures éclair conformes (sans défauts). Cette fonction était réservée aux clients Entreprise de Edge Impulse jusqu'au printemps 2025 mais a depuis été ouverte aux utilisateurs gratuits :

Visual Anomaly Detection - FOMO-AD OFFICIALLY SUPPORTED

Detect visual anomalies. Extracts visual features using a pre-trained backbone, and applies a scoring function to evaluate how anomalous a sample is by comparing the extracted features to the learned model. Does not require anomalous data.

Edge Impulse

L'apprentissage non supervisé présente l'avantage d'éviter la phase d'étiquetage que l'on a effectué ici dans le cas de l'apprentissage supervisé. Elle a cependant l'inconvénient de ne pas permettre une classification aisée des défauts de sortie car l'algorithme va soit classer les fermetures éclair comme des produits non conformes, soit regrouper les défauts par similitudes qui ne sont pas forcément celles souhaitées.

1.2.2 MISE EN FORME DES DONNEES

Cette phase ne présente pas de complexité particulière. On notera cependant :

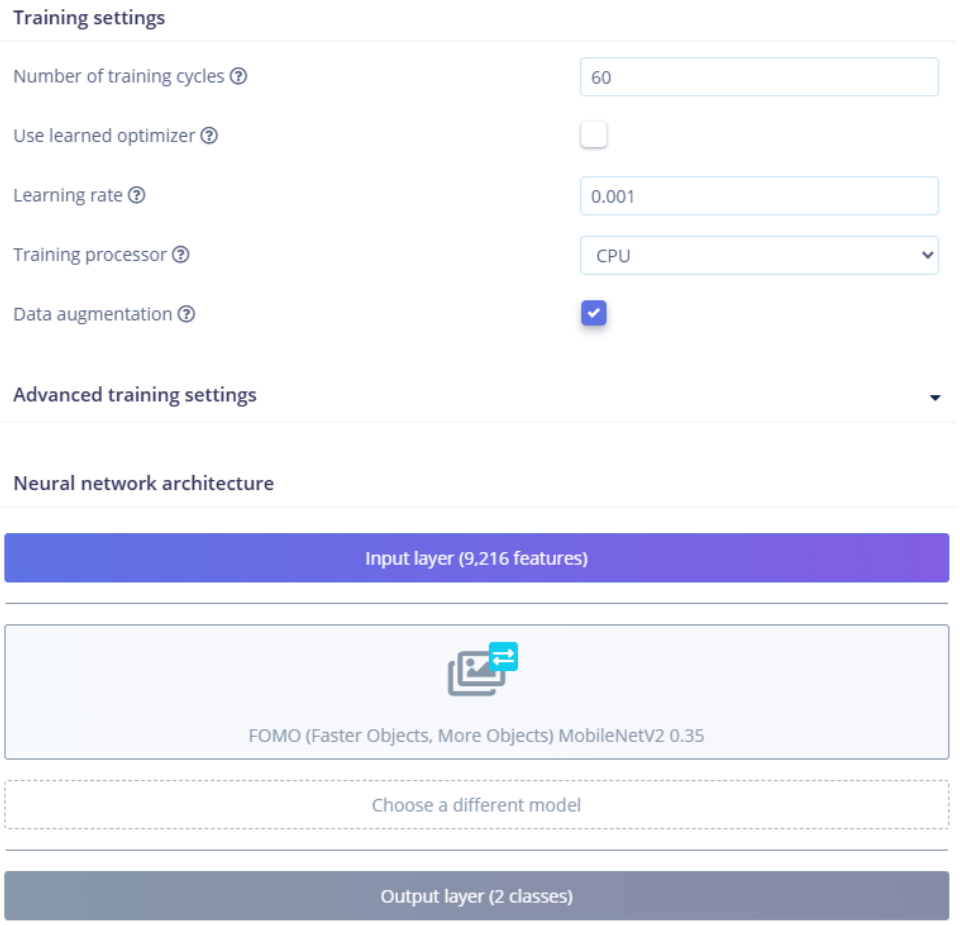
- Que le réseau de neurones travaille avec des images carrées et que l'on a plusieurs possibilités pour passer de l'image de base à des images carrées.

- Que le choix de raisonner sur des images en niveau de gris doit être guidé par la difficulté à détecter les classes sur les images : on peut conseiller de tester le fonctionnement sur des images en niveau de gris, puis si les performances ne sont pas satisfaisantes, de refaire l'entrainement sur des images en couleur (RGB).

1.2.3 Choix d'un modele

Cette étape présente toujours le point délicat du choix de la composition du réseau de neurones et du choix de ses paramètres d'apprentissage. On conserve ici les paramètres par défaut proposés par Edge Impulse, qui sont proposés à partir de similitudes avec d'autres projets similaires qui ont mené à des résultats satisfaisants.

On notera que le nombre d'attributs (features) est ici bien plus important que précédemment car on travaille avec des images :



1.2.4 Apprentissage

A l'issue de l'apprentissage, on obtient comme précédemment une matrice de confusion :

Last training performance (validation set)

F1 SCORE
94.7%

Confusion matrix (validation set)

	BACKGROUND	BROKEN	SPLIT
BACKGROUND	100.0%	0%	0.0%
BROKEN	0%	100%	0%
SPLIT	0%	0%	100%
F1 SCORE	1.00	1.00	0.93

Metrics (validation set)

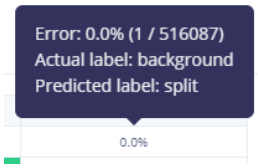
METRIC	VALUE
Precision (non-background)	0.90
Recall (non-background)	1.00
F1 Score (non-background)	0.95

On-device performance Engine: EON™ Compiler (RAM optimized)

INFERENCING TIME	PEAK RAM USAGE	FLASH USAGE
434 ms.	119.4K	90.9K

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

On note que les performances obtenues à l'issue de l'entrainement du modèle sont en apparence (voir test du modèle) très satisfaisantes, même si des erreurs d'identification sont présentes car le F1 Score du modèle n'atteint pas 100%. En observant en détail la matrice de confusion, on repère que l'erreur provient d'une zone d'image dans laquelle aucune classe n'a été étiquetée (background) pour laquelle l'algorithme a prédit une classe « split » (ci-contre).



On repère également dans cette matrice de confusion que les images du dataset (+ les images transformées par Data augmentation) ont été découpées en 516087 zones comprenant la classe background (aucune des deux classes à identifier), 2 zones étiquetées « broken » et 7 zones étiquetées « split ».

Bien évidemment, augmenter le nombre de défauts dans les images du dataset permettrait d'améliorer leur détection, mais générerait également un travail d'étiquetage plus conséquent. Ceci constitue un point bloquant dans le développement de ce modèle. On pourrait également travailler avec un apprentissage non supervisé pour se passer de la phase d'étiquetage, mais dans le cas présent cela ne permettrait pas d'identifier facilement le nombre de défauts de chaque classe pour essayer de les corriger.

<u>Remarque – consommation en ressources et consommation énergétique :</u>

Performances de ce modèle sur une architecture matérielle Cortex M4F 80MHz :

INFERENCING TIME	PEAK RAM USAGE	FLASH USAGE
434 ms.	119.4K	90.9K

Performances du modèle de classification de mouvement de la partie 1.1 sur une architecture matérielle Cortex M4F 80MHz :

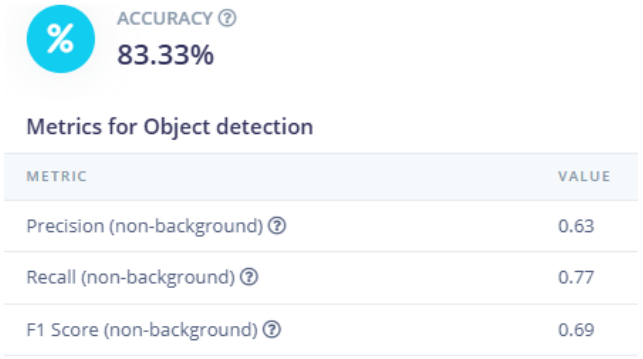
INFERENCING TIME	PEAK RAM USAGE	FLASH USAGE
1 ms.	1.4K	16.5K

On note du fait des observations ci-dessus une très importante différence au niveau de la consommation de ressources, et donc d'énergie pour l'inférence des deux modèles. Dès que l'on travaille avec des images, les demandes en calcul augmentent fortement. Ceci amène la question de l'utilité du travail sur les images : en a-t-on réellement besoin ou peut-on procéder différemment ? Par exemple, pour identifier un mode de marche sur une machine et si cela est possible, il sera

moins énergivore de travailler sur ses vibrations ou sur son bruit que de travailler avec des reconnaissances d'images.

1.2.5 TEST DU MODELE

A l'issue du test, on obtient les performances ci-contre. Pour les interpréter, il faut rappeler les notions de Precision et Recall, adaptées ici à l'exemple traité :



- Precision : Proportion de défauts "split" ou "broken" détectés par le modèle qui sont réellement présents dans l'image.

  ➔ Est-ce que les défauts que le modèle a signalés sont corrects ?

- Recall : Proportion de vrais défauts "split" ou "broken" présents dans les images que le modèle a effectivement détectés.
  ➔ Parmi tous les défauts annoncés par le modèle, combien sont vrais ?

Le F1-Score est ensuite obtenu à partir de la Precision et du Recall, comme la moyenne harmonique de ces deux grandeurs :

$$F_{1\ score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Illustration simplifiée :

Si l'on part d'un dataset contenant 10 images contenant chacune un défaut connu :

- 6 défauts « split »
- 4 défauts « broken »

Le modèle prédit :

- 5 « split », dont 4 corrects
- 6 « broken », dont 3 corrects

Les performances du modèle se calculent comme suit :

- Pour le défaut « split » :

  Precision = 4 / 5 = 80 % Recall = 4 / 6 = 66,7 %

- Pour le défaut « broken » :

  Precision = 3 / 6 = 50 % Recall = 3 / 4 = 75 %

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

On note que les performances d'identification obtenues ici ne sont pas si bonnes que celles obtenues sur les données d'entrainement. Ceci est dû en grande partie au faible nombre d'images contenant des défauts utilisé pour l'entrainement du modèle (même remarque qu'à la page précédente).

PARTIE 2 : DEPLOIEMENT SUR UN MICROCONTROLEUR

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

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

Le suivi des consignes du TP permet l'envoi direct des données acquises par le Wio terminal (ou une autre carte supportée) vers la plateforme Edge Impulse.

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

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

La démarche est rigoureusement identique à celle développée dans le paragraphe 1.1 et ne pose pas de difficultés particulières.

2.3.2 DEPLOIEMENT DU MODELE SUR UN MICROCONTROLEUR

Le suivi des instructions détaillées dans le TP permet la compilation puis le déploiement du modèle entrainé sur Edge Impulse vers le Wio Terminal. On obtient alors la classification de mouvements suivante :

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

La classification de mouvements fonctionne bien, avec les mêmes limites que celles évoquées dans la partie 1.1 (principalement, toutes les classes à identifier doivent être bien définies et le modèle doit être entrainé avec suffisamment de diversité parmi les classes). On note qu'avec le fonctionnement proposé ici, on alterne les phases d'acquisition de données et l'inférence du modèle d'IA menant à une prédiction de classe.

On pourrait imaginer un autre mode de fonctionnement effectuant une acquisition en continu, une définition glissante d'un échantillon sur les n dernières secondes d'acquisition, et une inférence du modèle d'IA se faisant en parallèle de l'acquisition et en quasi temps continu. Un tel fonctionnement est beaucoup plus gourmand en ressources et incompatible avec les capacités de calcul du Wio Terminal.

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.

Pas de correction pour cette partie, mais on pourra s'appuyer sur les tutoriels proposés par Seeed Studio.