TensorFlow Lite pour microcontrôleurs

par un débutant, pour les débutants

Jean-François Simon (Elektor)

TensorFlow Lite pour microcontrôleurs permet de faire tourner des modèles d'apprentissage automatique (*Machine Learning*, ou *ML*) sur des plateformes aux ressources limitées. Voyons cela de plus près et utilisons-le avec Edge Impulse pour la reconnaissance vocale sur un Arduino Nano 33 BLE Sense.

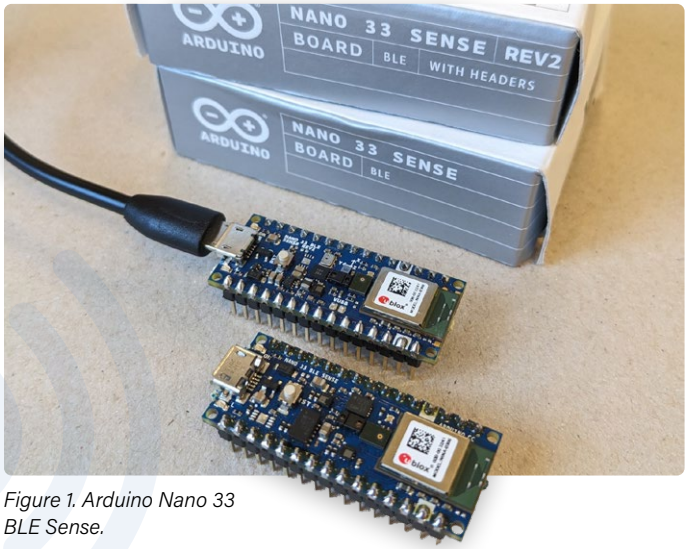

Figure 1. Arduino Nano 33 BLE Sense.

L'intelligence artificielle (IA) et l'apprentissage automatique (ML) sont les nouveaux mots à la mode, parfois utilisés de manière un peu abusive. Facebook, Amazon, Google et bien d'autres utilisent des systèmes d'apprentissage automatique pour vous proposer des contenus les plus adaptés à vos goûts et à vos habitudes. ChatGPT est un autre exemple de service très spectaculaire et populaire utilisant la ML. Le point commun de ces entreprises est qu'elles ont accès à des serveurs dotés d'une énorme puissance de calcul pour entraîner les modèles en traitant de très grands volumes de données, et pour répondre de manière fluide aux requêtes d'un grand nombre d'utilisateurs.

C'est toutefois en train de changer avec l'émergence de l'IA « en périphérie », c'est-à-dire l'utilisation d'algorithmes d'intelligence artificielle en périphérie du réseau, à savoir plus près de la source de données et loin d'un serveur. Cela permet d'analyser les données et de prendre des décisions en temps réel, en réduisant la latence et les besoins en bande passante. Bien que la notion de réseau soit souvent mise en avant, le concept fonctionne également sans réseau du tout - par exemple, sur une modeste carte à microcontrôleur, qui n'est pas nécessairement connectée à Internet.

TensorFlow Lite pour microcontrôleurs

Une évolution intéressante s'est produite dans ce domaine il y a quelques années avec l'apparition de TensorFlow Lite pour microcontrôleurs (TFLite Micro [1]. Il s'agit d'une version allégée de TensorFlow, un framework d'apprentissage machine open-source développé par Google. Cette version est conçue pour exécuter des modèles de ML sur des microcontrôleurs aux ressources limitées. Alors, pouvez-vous utiliser TFlite Micro sur votre carte Arduino ? Oui, mais pas sur tous les Arduino. Il est écrit en C++ 17 et nécessite une plateforme 32 bits, ainsi que quelques kilooctets de RAM. Il peut être utilisé avec de nombreux microcontrôleurs Arm Cortex-M, ainsi qu'avec l'ESP32. La liste complète des plateformes compatibles est disponible sur [1]. Ainsi, si le vénérable Arduino Uno n'est pas à la hauteur, l'Arduino Nano 33 BLE Sense (**figure 1**) peut être utilisé. Cette carte est en fait idéale pour expérimenter, car elle est puissante et contient de nombreux capteurs : un accéléromètre, un capteur d'humidité, de température, de couleur et d'intensité de la lumière, un capteur de pression et un microphone.

Bien que cette carte Arduino soit puissante, elle ne l'est pas encore assez pour entraîner le modèle directement sur la carte. Dans la plupart des projets de ML basés sur des microcontrôleurs, la méthode habituelle consiste à préparer les données sources et à entraîner un modèle sur une machine puissante, telle que votre PC ou un serveur distant. Le modèle entraîné obtenu prend la forme d'un fichier binaire, qui doit être converti ultérieurement en un fichier d'en-tête en langage C. Enfin, un programme Arduino peut être écrit en utilisant les fonctions fournies dans la bibliothèque TFLite Micro et compilé avec l'IDE Arduino.

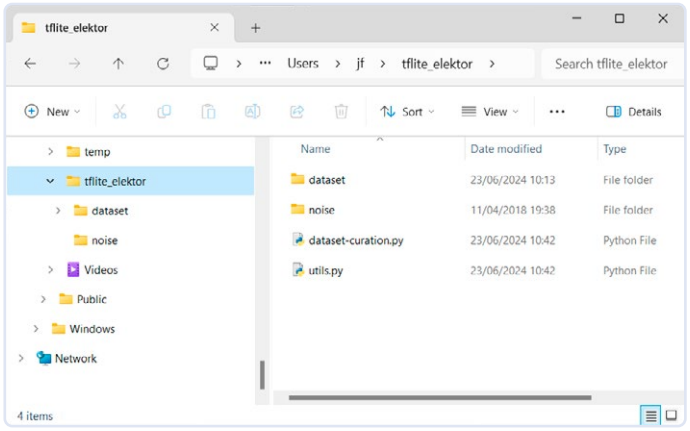

Figure 2. Les fichiers et dossiers avant l'exécution du script Python.

Pour ceux qui aiment tout faire par eux-mêmes, voyez la documentation officielle de TensorFlow Lite [2]. J'ai également trouvé des articles intéressants publiés par DigiKey [3]. Ils recommandent d'utiliser un ordinateur sous Linux doté de Python, puis d'installer, entre autres, TensorFlow, Keras, Anaconda, Jupyter Notebook et d'autres paquets. Une autre solution consiste à exécuter le code Python dans Google Colab [4], une plateforme gratuite basée sur le cloud qui permet aux utilisateurs d'écrire et d'exécuter du code Python dans un environnement en ligne. En tant que novice, j'ai trouvé la documentation de TensorFlow particulièrement difficile à suivre. Il faut également avoir une bonne compréhension des réseaux neuronaux pour pouvoir faire quoi que ce soit de fonctionnel, ce qui est vite décourageant.

Exemples simples

Les tutoriels sur Internet ont tendance à montrer toujours un peu les mêmes choses, dont certaines manquent d'utilité pratique pour être vraiment stimulantes. Par exemple, il est souvent montré comment entraîner un modèle à produire une valeur de sortie correspondant à une approximation du sinus de la valeur d'entrée. Cette méthode utilise, bien entendu, les valeurs précalculées de la fonction sinus comme ensemble de données d'apprentissage. Ainsi, une fois correctement entraîné, le modèle peut donner une valeur approximative de sin(x) à sa sortie, pour une valeur d'entrée x comprise entre 0 et 2π. Cela, bien sûr, sans utiliser de fonction sinus implémentée mathématiquement. Évidemment, il s'agit probablement de la manière la plus absurde et la moins pratique de calculer un sinus, en particulier sur un microcontrôleur où la puissance de calcul est limitée.
Un autre exemple, plus utile, est celui de la reconnaissance vocale. Le microcontrôleur peut ainsi écouter ce qui se passe dans son environnement à l'aide d'un microphone, discerner quelques mots (par exemple, *oui* et *non*, ou *chat* et *chien*, etc.) et déclencher diverses actions. Cet article, écrit par un débutant pour les débutants, reste simple et montre l'utilisation de la reconnaissance vocale sur un Arduino Nano 33 BLE Sense.

Reconnaissance vocale

À cet effet, le Google Speech Command Dataset sera utilisé. Il s'agit d'un dossier contenant 65 000 échantillons audio d'une seconde ; au total, une trentaine de mots différents prononcés par des milliers de personnes différentes. Pour entraîner le modèle, j'utiliserai Edge Impulse [5]. Il s'agit d'une plateforme qui permet de créer, d'entraîner et de déployer des modèles d'apprentissage automatique sur du matériel embarqué, tel que des microcontrôleurs, en mettant l'accent sur la facilité d'utilisation, sans nécessiter trop de programmation. Edge Impulse utilise TensorFlow Lite pour Microcontrôleurs et fournit un moyen facile de déployer le modèle et la bibliothèque TFLite sur la carte Arduino elle-même, ce qui est très pratique.
Pour commencer, vous aurez besoin de quelques échantillons audio. Créez un dossier, qui sera votre dossier de travail. J'ai appelé le mien *tflite_elektor*. Téléchargez le Google Speech Command Dataset [6]. Assurez-vous d'avoir une bonne connexion Internet, car le fichier pèse 2,3 Go. Comme il s'agit d'un fichier avec une extension *.tar.gz*, il est compressé deux fois. Utilisez 7-Zip ou équivalent (je ne recommande pas l'utilitaire intégré de Windows pour traiter des fichiers aussi volumineux) pour obtenir le fichier *.tar* à l'intérieur, puis décompressez son contenu. Le résultat est un dossier *speech_commands_v0.02*. Placez ce dossier dans votre dossier de travail. Vous pouvez renommer le dossier *speech_commands_v0.02* pour lui donner un nom plus simple, dans mon cas : *dataset*.

Préparation des données

Ensuite, il faut préparer les données. Pour cela, je suggère d'utiliser l'excellent script Python développé par Shawn Hymel, qu'il propose généreusement sous licence libre. Téléchargez les fichiers *dataset-curation.py* et *utils.py* depuis son dépôt GitHub [7] et enregistrez-les dans votre dossier de travail. Ce script nécessite que le dossier *_background_noise_* à l'intérieur du dossier *dataset* soit séparé des mots-clés. Glissez-déposez donc ce dossier en dehors du dossier *dataset* pour le placer dans votre dossier de travail. Vous pouvez également le renommer : *noise*. Votre dossier de travail contient maintenant les deux dossiers *dataset* et *noise* ainsi que les deux fichiers Python (**figure 2**).
Le script Python facilite grandement l'utilisation de l'énorme quantité de données contenue dans le dossier précédemment téléchargé. En outre, comme vous le verrez plus loin, le script est flexible et peut être utilisé avec d'autres données audio que celles-ci, y compris avec des fichiers audio que vous avez enregistrés vous-même. Il ne serait pas pratique de devoir télécharger plusieurs gigaoctets de fichiers vers les serveurs d'Edge Impulse. Pour commencer, choisissez un ou plusieurs mots-clés, qui seront les mots cibles que l'Arduino sera chargé de détecter. Pour cet exemple, j'ai choisi le mot *zéro*. Le script va créer un ensemble de dossiers : un dossier pour chaque mot-clé cible, donc dans ce cas un seul dossier nommé *zero*, ainsi qu'un dossier *_noise*, contenant du bruit aléatoire, et un dossier *_unknown* contenant des mots aléatoires autres que les mots-clés cibles.
Le script ajoute du bruit de fond dans les fichiers audio contenant les mots-clés, via un ré-échantillonnage. Il crée d'abord les dossiers nécessaires, puis extrait de petits échantillons de bruit de fond, qui sont ensuite mélangés avec des échantillons de mots-clés cibles et de mots-clés non cibles. Cela améliorera la résistance du modèle aux bruits de fond ; le résultat est un ensemble de données préparées, de taille beaucoup plus réduite (environ 140 mégaoctets), qu'Edge Impulse peut facilement importer.

Utilisation de Python

Le code a été testé avec Python 3.7. Pour gérer plusieurs environnements Python de différentes versions, avec différents paquets installés, il est possible d'utiliser Anaconda [8], qui facilite la création

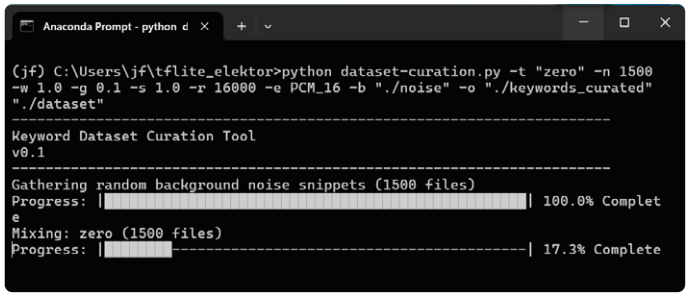

Figure 3. Exécution du script Python.

d'une installation « toute neuve » de la version souhaitée. Ici, je crée un nouvel environnement appelé *jf* :

```
conda create -n jf python=3.7
```

Ensuite, vous devez installer les paquets *librosa*, *numpy* et *soundfile* :

```
python -m pip install librosa numpy soundfile
```

Le paquet *shutil* est également nécessaire, mais il est normalement inclus avec Python 3.7.

Depuis la ligne de commande Anaconda ou celle de votre système, naviguez jusqu'à votre répertoire de travail et exécutez le script à l'aide de la commande :

```
python dataset-curation.py -t "zero" -n 1500 -w 1.0 -g 0.1 -s 1.0 -r 16000 -e PCM_16 -b "./noise" -o "./keywords_curated" "./dataset"
```

Et attendez quelques minutes que l'opération se termine (**figure 3**).

Jetons un rapide coup d'œil sur les arguments pris par le script :

`-t` sert à lister les mots-clés cibles. Ici, j'utiliserai `-t "zero"`.

`-n` est le nombre d'échantillons de sortie par catégorie. 1500 est un bon point de départ.

`-w` et `-g` sont les volumes sonores du mot parlé et du bruit de fond, respectivement. `-w 1.0 -g 0.1` sont des valeurs recommandées.

`-s` et `-r` sont la longueur de l'échantillon (1 s) et le taux de rééchantillonnage (16 kHz). Utilisez `-s 1.0 -r 16000`.

`-e` est le nombre de bits pour l'échantillonnage, ici le PCM 16 bits est utilisé.

`-b` est l'emplacement du dossier du bruit de fond, `-o` est celui du dossier de sortie et enfin, le dernier argument non étiqueté est la liste des dossiers d'entrée. Ici, il s'agit du dossier *dataset*.

Lorsque le script a fini son traitement, il devrait avoir créé un dossier *keywords_curated* contenant trois dossiers : *_noise*, *_unknown* et *zero*. (**figure 4**)

Importation dans Edge Impulse

L'étape suivante consiste à importer ces fichiers dans Edge Impulse. Rendez-vous sur leur site web et créez un compte si vous n'en avez pas déjà un. Après avoir ouvert une session, créez un nouveau projet. Dans le menu de gauche, naviguez vers *Data Acquisition*, puis cliquez sur *Add Data* et *Upload Data*. Cochez *Select a folder* et choisissez le premier dossier, par exemple *_noise*.

Veillez à cocher l'option *Automatically split between training and testing*. De cette façon, Edge Impulse utilisera d'abord 80 % des échantillons téléchargés pour entraîner le modèle. Ensuite, nous pourrons tester les performances du modèle entraîné en utilisant des données qu'il n'a jamais vues auparavant ; les 20 % restants sont réservés à cet effet.

Cochez également l'option *Label: infer from filename* afin que Edge Impulse reconnaisse, par le nom du fichier, les échantillons contenant le(s) mot(s) à reconnaître ainsi que ceux contenant du bruit.

Enfin, cliquez sur le bouton *Upload data* dans le coin inférieur droit et attendez que le transfert soit terminé. Répétez l'opération pour les deux dossiers restants *_unknown* et *zero*.

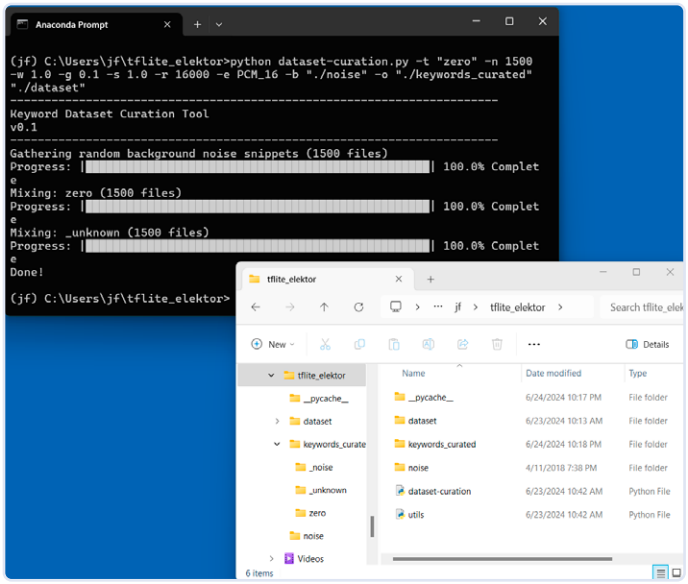

Figure 4. Après l'exécution du script Python.

Une fois le transfert terminé, retournez dans *Data Acquisition* pour visualiser tous les échantillons téléchargés. Assurez-vous que les fichiers sont bien répartis en 80% de données d'entraînement et 20% de données de test, et que les étiquettes ont été correctement lues (**figure 5**).

Ensuite, il est nécessaire d'ajouter un *Processing Block*. Dans Edge Impulse, il s'agit d'un composant utilisé pour transformer les données brutes en un format adapté à l'entraînement et à l'inférence des modèles d'apprentissage automatique. Sous ce simple « bloc » se cachent plusieurs étapes plus complexes, comme le prétraitement des données d'entrée, l'extraction des *caractéristiques* (voir ci-dessous), des étapes optionnelles telles que des transformées de Fourier, etc.

Dans le contexte de la ML, les *caractéristiques* ou *features* sont des propriétés distinctes et quantifiables des données observées. Ici, les caractéristiques à extraire sont les *Mel-Frequency Cepstral Coefficients* (MFCC) [9], qui sont couramment utilisés dans le traitement des signaux audio et la reconnaissance vocale. Ils représentent le spectre de puissance à court terme d'un signal sonore sur une échelle non linéaire de fréquence.

Ainsi, dans le menu de gauche cliquez sur *Impulse Design* puis sur le bouton *Add a Processing Block*. Sélectionnez la première option, *Audio (MFCC)*, en cliquant sur *Add* à droite. Ensuite, cliquez sur le

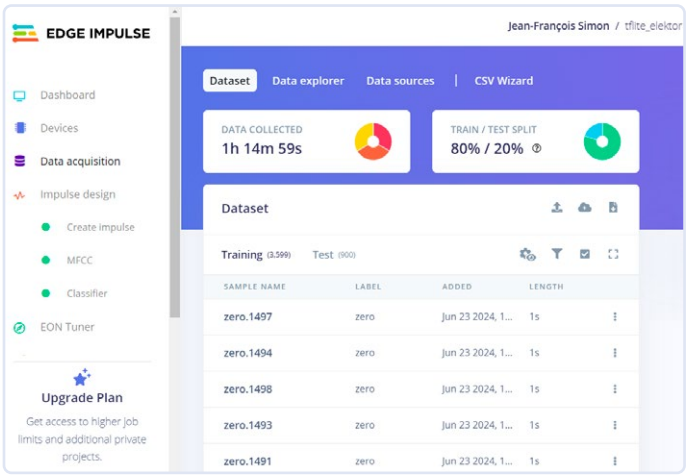

Figure 5. Les échantillons audio sont correctement stockés par Edge Impulse.

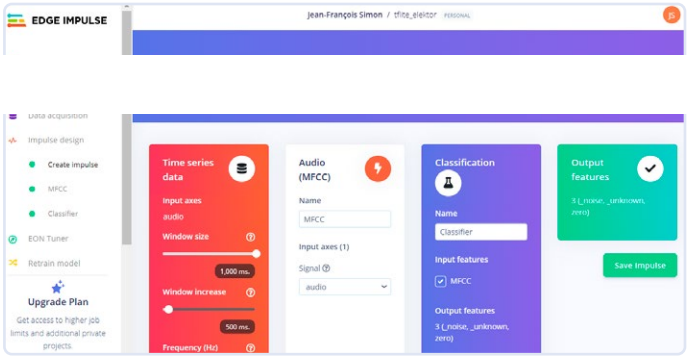

Figure 6. Configuration du modèle.

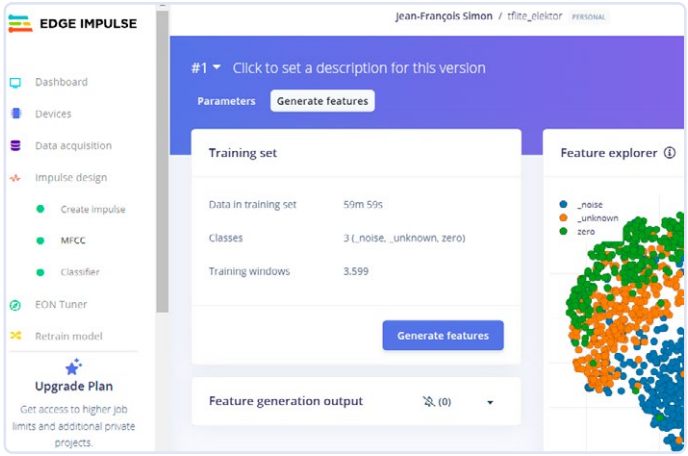

Figure 7. La section Generate Features, où les données audio sont traitées.

bouton *Add a Learning Block* (Ajouter un bloc d'apprentissage) et choisissez la première option, *Classification*, qui est celle recommandée. Enfin, cliquez sur *Save Impulse* à droite (**figure 6**).

Entraînement du modèle

Dans le menu de gauche, sous *Impulse Design*, sélectionnez *MFCC*. Naviguez jusqu'à l'onglet *Generate Features* et cliquez sur *Generate Features* (**figure 7**). Attendez que la génération des caractéristiques soit terminée. Ceci fait, allez dans la section *Classifier*, située juste en dessous de *MFCC* dans le menu de gauche. Dans le coin supérieur droit, cliquez sur *target* et sélectionnez *Arduino Nano 33 BLE Sense*. Vous pouvez ajuster les paramètres du réseau de neurones, mais les paramètres par défaut sont, sans surprise, meilleurs que tout ce que j'aurais pu faire moi-même. Notez que vous pouvez éditer le réseau neuronal à l'aide de l'outil graphique ou passer en mode expert via le menu contextuel si vous êtes familier avec Keras. Pour cet exemple, cliquez simplement sur *Start Training* en bas de la page pour commencer à entraîner le modèle. Une fois l'entraînement terminé, examinez les résultats dans le cadre *Model* en bas à droite. Vous verrez un score général de précision, et 90 % est considéré comme un très bon score ; ici, le système a obtenu 92,8 % (**figure 8**).

Il existe également une matrice, appelée *matrice de confusion*, qui vérifie les performances du modèle. Les lignes représentent les étiquettes réelles et les colonnes les étiquettes prédites. Les nombres situés sur la diagonale, où l'étiquette prédite correspond à l'étiquette réelle, devraient être beaucoup plus élevés que les autres valeurs. Ici, la diagonale affiche 98,8 %, 87 % et 92,8 %, ce qui devrait être suffisant. Un test plus difficile consiste à évaluer le modèle en lui fournissant des données qu'il n'a jamais vues auparavant. Pour ce faire, accédez à la section *Model testing* dans le menu de gauche. Cliquez sur *Classify All* et laissez l'opération se dérouler. Dans le cadre *Results*, en bas, le score est inférieur de quelques pourcents au score précédent, mais c'est normal. Ici, on obtient 90,56 % (**figure 9**).

Déploiement pour Arduino

Passons maintenant à la page *Deployment*. Edge Impulse offre plusieurs options pour empaqueter le modèle : une bibliothèque C++ générique pour un usage général sur microcontrôleurs, Cube MX pour les composants STM32, WebAssembly pour les environnements JavaScript et bien d'autres encore. Cliquez sur *Search deployment options* et sélectionnez *Arduino library*. Cliquez ensuite sur le bouton *Build* en bas de la page. Quelques secondes plus tard, votre navigateur téléchargera un fichier ZIP contenant la bibliothèque Arduino.

J'utilise l'IDE Arduino version 1.8.19. Ouvrez l'Arduino IDE et connectez votre Arduino Nano 33 BLE Sense à votre ordinateur. Si c'est la première fois que vous utilisez votre Nano 33 BLE, l'IDE vous proposera de télécharger un paquet appelé *Arduino Mbed OS Nano Boards package*, qui est effectivement nécessaire. Ensuite, vous pouvez ajouter la bibliothèque en utilisant la technique habituelle, en cliquant sur *Croquis, Inclure une bibliothèque, Ajouter la bibliothèque ZIP* et en sélectionnant le fichier .zip que vous venez de télécharger depuis Edge Impulse. Ensuite, allez dans *Fichier, Exemples* et localisez la bibliothèque que vous venez d'installer. Il se peut que vous deviez recharger l'IDE Arduino pour qu'elle apparaisse. Le nom doit correspondre au nom de votre projet Edge Impulse, c'est donc *tflite_elektor_inferencing* dans mon cas.

Notez qu'il y a deux dossiers séparés, *nano_ble33_sense* et *nano_ble33_sense_rev2* (**figure 10**). L'exemple *microphone_continuous* utilisé ici n'apparaît que dans le premier, mais je l'ai testé avec succès sur les deux révisions de carte. En revanche, vous devrez probablement choisir la bonne version en fonction de la carte dont vous disposez si vous voulez expérimenter avec les autres exemples de programmes qui utilisent l'accéléromètre intégré. Ouvrez l'exemple *microphone_continuous*.

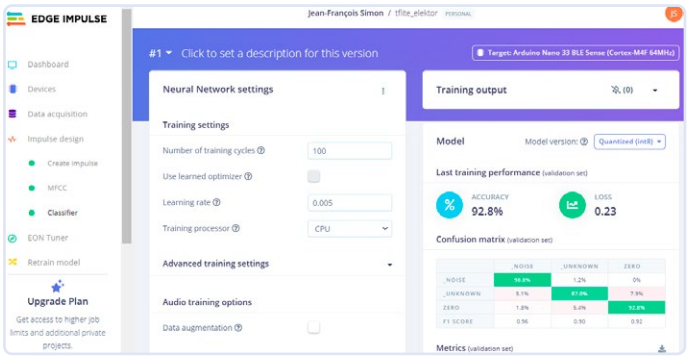

Figure 8. Le modèle a terminé son entraînement.

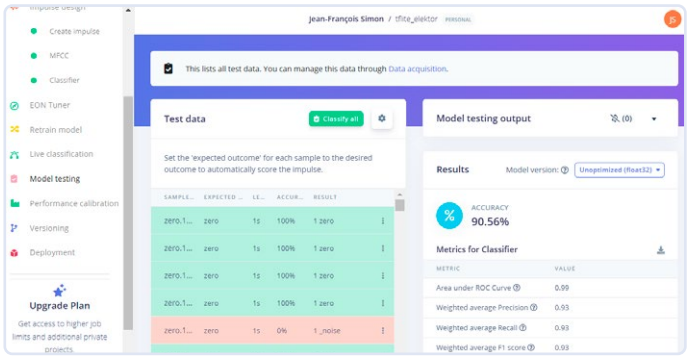

Figure 9. Test du modèle sur de nouvelles données.

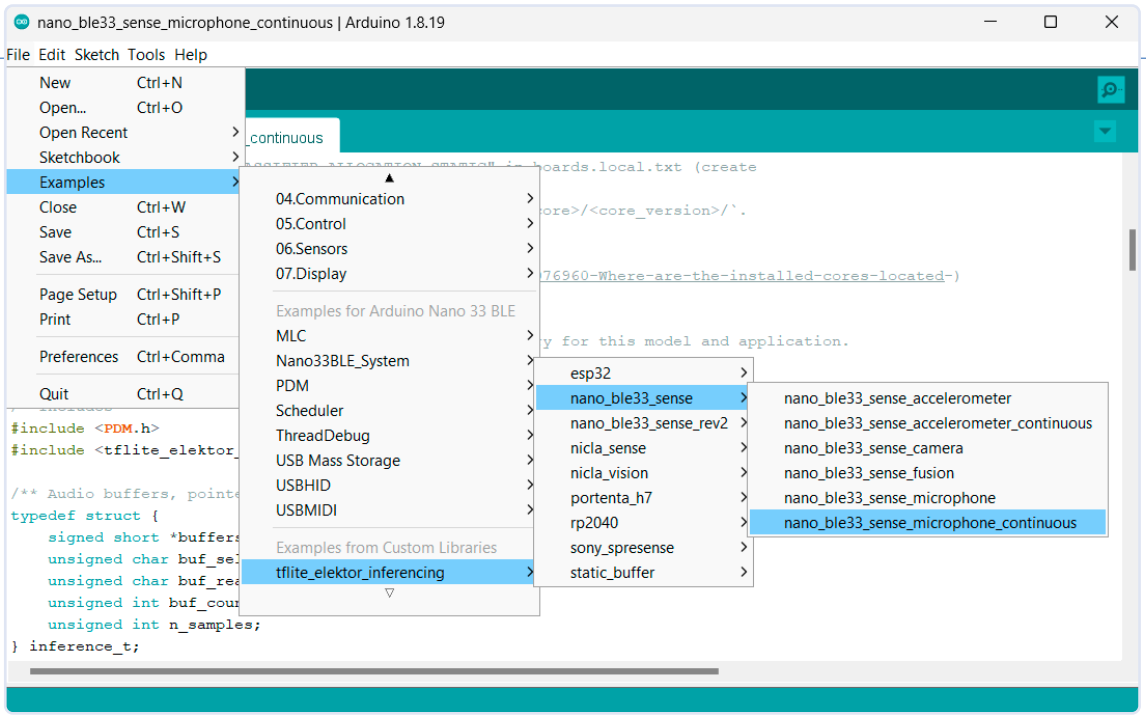

Figure 10. Utilisation de l'exemple intégré microphone_continuous.

Si vous le souhaitez, vous pouvez explorer le programme exemple pour comprendre comment tout est mis en place et quelles fonctions sont appelées pour l'inférence. Dans la boucle, le microcontrôleur attend que la mémoire tampon du microphone se remplisse, puis appelle la fonction `run_classifier_continuous` pour effectuer l'inférence. Les résultats ne sont affichés sur le Moniteur Série qu'une fois par seconde. Le code de la bibliothèque fournie n'est parfois pas facile à suivre, mais essayer de voir ce qu'il y a sous le capot peut être un exercice enrichissant.

Programmation de la carte

Dans le menu Outils de l'IDE Arduino, assurez-vous que la bonne carte (Nano 33 BLE Sense) est sélectionnée, ainsi que le bon port COM, que vous pouvez vérifier à l'aide du Gestionnaire de périphériques si vous utilisez Windows. Cliquez sur le bouton *Téléverser* et attendez ! Gardez à l'esprit que la compilation du projet prend un certain temps car la bibliothèque, qui contient les fonctions TensorFlow Lite pour l'inférence ainsi que le modèle créé sur Edge Impulse au format binaire, est assez conséquente. Une fois

Un sujet complexe, et des options plus simples

Pourquoi ne voit-on pas une plus grande variété de projets et d'exemples utilisant la ML sur de petits microcontrôleurs ? À mon avis, il y a plusieurs raisons.

- Pour entraîner un modèle, vous avez besoin d'un grand volume de données pré-classifiées. Pour la fonction sinus, c'est facile : les résultats attendus sont connus à l'avance. Pour les mots-clés anglais, vous pouvez utiliser le jeu de données gratuit Google Speech Command Dataset. Mais pour faire autre chose que de la reconnaissance de mots, il faudra acquérir une grande quantité de données adaptées à votre projet, ce qui peut constituer une difficulté majeure.

- L'apprentissage automatique est un domaine complexe à aborder. Pour sortir des sentiers battus, il faut être très à l'aise avec Python et Linux et avoir une bonne compréhension de la théorie de la ML et des réseaux neuronaux. Personnellement, j'ai trouvé cela très difficile d'accès, et je n'ai clairement pas les qualifications ni l'expérience requises pour aller plus loin que cet exemple de base.

- La documentation existante pour TensorFlow et TensorFlow Lite s'adresse principalement aux développeurs, et il semblerait que nombre d'entre eux soient davantage intéressés par les plateformes plus puissantes, telles que le Raspberry Pi 5 équipé d'un accélérateur d'IA externe, ou le Jetson Nano de Nvidia. Cela dit, Pete Warden, un contributeur clé de TensorFlow et maintenant PDG de Useful Sensors, a récemment fait une mise à jour majeure de la version TFLite Micro pour le Raspberry Pi Pico, qui a été créée en 2021 mais assez peu utilisée depuis. Espérons que cela revitalise le projet et incite les gens à l'utiliser !

Si vous souhaitez simplement ajouter quelques fonctions de contrôle vocal à votre projet, sans utiliser les processus complexes décrits dans cet article, vous pouvez envisager trois autres options : le Voice Recognition Module V3d'Elechouse, le Speech Recognizer de Grove et le Offline Language Learning Voice Recognition Sensor de DFRobot. Vendus respectivement 30, 20 et 17 euros, ces modules sont un peu plus chers que ce à quoi on s'attendrait, mais ils peuvent constituer une solution simple à mettre en œuvre. Ces modules n'utilisent pas le même principe que celui décrit dans cet article ; vous pouvez les entraîner à reconnaître votre propre voix en répétant les commandes un petit nombre de fois. Je ne les ai jamais utilisés moi-même, mais ils semblent assez efficaces, et vous trouverez de nombreuses vidéos de démonstration sur YouTube.

```
Predictions (DSP: 76 ms., Classification: 6 ms., Anomaly: 0 ms.):
    _noise: 0.15234
    _unknown: 0.84766
    zero: 0.00000
Predictions (DSP: 76 ms., Classification: 6 ms., Anomaly: 0 ms.):
    _noise: 0.98438
    _unknown: 0.01172
    zero: 0.00391
```

Figure 11. L'Arduino écoute les bruits aléatoires dans la pièce.

```
Predictions (DSP: 76 ms., Classification: 6 ms., Anomaly: 0 ms.):
    _noise: 0.00000
    _unknown: 0.00391
    zero: 0.99609
```

Figure 12. Le mot a été détecté avec une excellente certitude.

le programme compilé, vous verrez qu'il utilise environ 171 kilooctets de mémoire Flash et environ 47 kilooctets de RAM pour les variables globales.

Ça marche !

Ouvrez maintenant le Moniteur Série pour observer la sortie. Chaque seconde, il donne trois nombres qui sont les probabilités qu'un motif donné ait été détecté, parmi : un bruit aléatoire, un mot qui n'est pas *zéro* et enfin le mot *zéro*. La **figure 11** est un exemple où rien de spécial ne se produit. Si je prononce le mot « zéro » relativement près de la carte Arduino, le troisième score atteint une valeur très élevée, presque 100 % (**figure 12**).

C'est un bon début ! L'étape suivante consisterait à faire en sorte que la carte Arduino fasse quelque chose d'utile avec cette information. Je suis sûr que vous trouverez des applications intéressantes pour envoyer des commandes vocales à des gadgets utilisant un Arduino. Le processus décrit ci-dessus et le script Python conçu par Shawn Hymel peuvent également être utilisés pour détecter plus d'un mot. Le nombre maximum sera limité par l'espace de stockage dans la mémoire Flash et la puissance de calcul disponibles sur l'Arduino. Dans le code, la ligne `#define EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW 4` nous indique que chaque fenêtre d'une seconde est divisée en quatre tranches de 250 ms, et la sortie dans le Moniteur Série nous indique que le temps utilisé par le programme est de 76 + 6 = 82 ms par tranche de 250 ms, ce qui correspond approximativement à 33% d'utilisation du CPU. Il reste donc du temps de traitement disponible pour ajouter votre propre programme.

Pour aller plus loin

Par souci de simplicité, j'ai utilisé l'un des mots déjà disponibles dans le Google Speech Command Dataset. Pour entraîner un modèle à repérer un mot qui ne fait pas partie de cet ensemble, vous devez enregistrer un grand nombre d'échantillons audio avec le mot prononcé, de préférence par un grand nombre de personnes avec des voix, des âges et des intonations différents. Alors que le dossier Google contient des milliers d'échantillons par mot, lorsque vous enregistrez vous-même des mots personnalisés, cinquante à cent échantillons peuvent constituer un bon début. Bien entendu, je n'ai fait qu'effleurer le sujet avec cet exemple simple. Je recommande aux personnes intéressées de l'explorer plus en profondeur ! Avez-vous une idée en tête pour un projet utilisant la ML ?

240357-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (jean-francois.simon@elektor.com), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Jean-François Simon a une passion de longue date pour l'électronique et s'intéresse à des sujets aussi variés que la conception de circuits, le test et la mesure, le prototypage, la radio logicielle (SDR), et plus encore. Il aime créer, modifier et améliorer ses outils et autres systèmes. Il a une formation d'ingénieur et aime aussi la mécanique, l'usinage et tout ce qui est technique. Jean-François a rejoint l'équipe d'Elektor en 2023.

Produits

- **Arduino Nano 33 BLE Sense Rev2 avec connecteurs**
  www.elektor.fr/20404

LIENS

[1] TensorFlow Lite pour microcontrôleurs : https://www.tensorflow.org/lite/microcontrollers
[2] TFLite Micro sur GitHub : https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples
[3] Digikey, TensorFlow Lite Tutorial Part 1 : Wake Word Feature Extraction : https://tinyurl.com/bdf4dkfm
[4] Google Colab : https://colab.research.google.com/
[5] Edge Impulse : https://edgeimpulse.com/
[6] Google Speech Command Dataset : http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz
[7] Scripts Python de Shawn Hymel : https://github.com/ShawnHymel/ei-keyword-spotting
[8] Anaconda : https://www.anaconda.com/download
[9] MFCCs sur Wikipedia : https://en.wikipedia.org/wiki/Mel-frequency_cepstrum