Processus d'entraînement et d'implémentation d'un modèle d’apprentissage profond pour une tâche de détection d’événements sonores par Arduino

Table des matières :

I. Entraînement d'un modèle TensorFlow pour une tâche de détection automatique par Arduino

1. Constitution des jeux de données

1.1.Créer ou récupérer des jeux de données correspondants à des audios de chaque classe à prédire.

- Classe sirènes : sons d'internet (youtube, banque de sons)
- Classe autres : sons d'internet (youtube, banque de sons)

1.2.Enregistrer les sons de ces jeux de données avec le microphone Arduino au format souhaité (fréquence, longueur des enregistrements).

- Classe sirènes : 9 min d'enregistrements de 1 sec à 8kHz
- Classe autres : 9 min d'enregistrements de 1 sec à 8kHz

1.3.1.Enregistrer avec le microphone Arduino des bruits de fonds (background noise) qui peuvent souvent se produire et perturber les audios de chaque classe dans le but d'augmenter les jeux de données du microphone Arduino.

- Enregistrement des sons du robot

1.3.2.Enregistrer également ces background noises avec un son des classes à prédire pour déterminer le Signal-to-Noise Ratio (SNR)

- Enregistrement des sons du robot avec un son de sirène en même temps pour déterminer le SNR
- Déterminé le SNR en augmentant avec Audiomentation le son de sirène avec le son du robot correspondant aux sons de l'enregistrement précédent dans le but d'avoir le même équilibre sonore que l'enregistrement précédent (comparé avec Audacity) : SNR = -44.7dB

2. Création de jeux de tests

2.1.Sélectionner aléatoirement un même pourcentage d'audios des jeux de données Arduino.

- 20% d'audios pris pour le test (10% classe sirènes, 10 % classe autres)

2.2.Dupliquer ces jeux de test et augmenter les copies avec les background noises collectés en fonction du SNR déterminé au préalable.

- 1 jeu de test sirènes/autres sans augmentation
- 1 jeu de test sirènes/autres augmenté à 50%
- 1 jeu de test sirènes/autres augmenté à 100%

3. Traitement du signal audio

3.1.Changer l'échelle des données en normalisant ou standardisant les données.

- Normalisation des audios avec min-max normalization

3.2.Traiter le signal audio normalisé pour extraire les caractéristiques qui nous intéresse.

- Calcul des magnitudes du signal, puis calcul de 5 frames de 32 coefficients mel

4. Entraînement du modèle

4.1.Concevoir un modèle (Tensorflow pour l'instant) d'apprentissage profond adapté aux caractéristiques que l'on veut utiliser.

- Convolutional Neural Network (Conv2D (ReLu) → MaxPooling2D → Conv2D (ReLu → MaxPooling2D → FullyConnected 10 (ReLu) → FC 1 (Sigmoid)) avec pour entrées 5 frames de 32 coefficients mels

4.2.Entraîner le modèle avec des jeux de données augmentés par rapport au jeu de données de base et garder le modèle le plus satisfaisant.

- Entraînement sur jeu de train non augmenté : précision sur jeu de test
  - pas augmenté : 95%
  - *augmenté à 50%* : 92.3%
  - augmenté à 100% : 91%
- Entraînement sur jeu de train augmenté à 50% : précision sur jeu de test
  - pas augmenté : 95.2%
  - *augmenté à 50%* : **94.47%**
  - augmenté à 100% : 93.8%

II. Importation et utilisation du modèle TensorFlow sur Arduino

5. Importation du modèle sur Arduino avec Edge Impulse

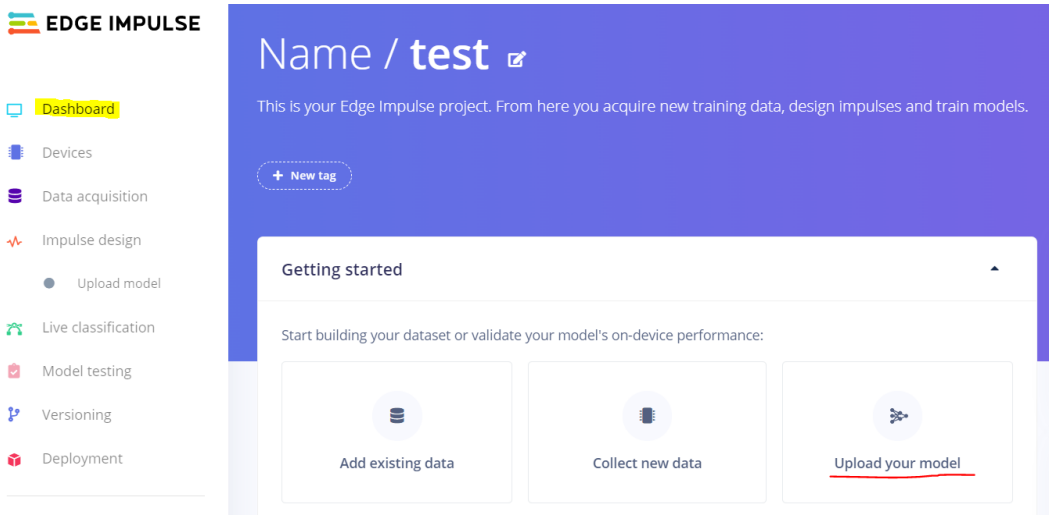
5.1.Enregistrer le modèle TensorFlow dans un des formats acceptés par la plateforme Edge Impulse :

TensorFlow SavedModel (`saved_model.zip`), ONNX model (`.onnx`) or TensorFlow Lite model (`.tflite`)

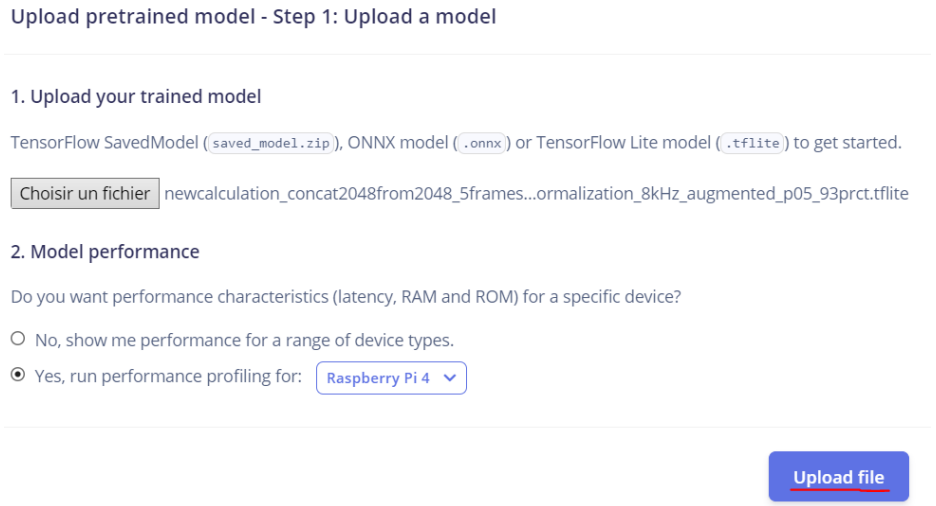
- Enregistrement du modèle au format .tflite

5.2.Créer une library Arduino du modèle avec Edge Impulse :

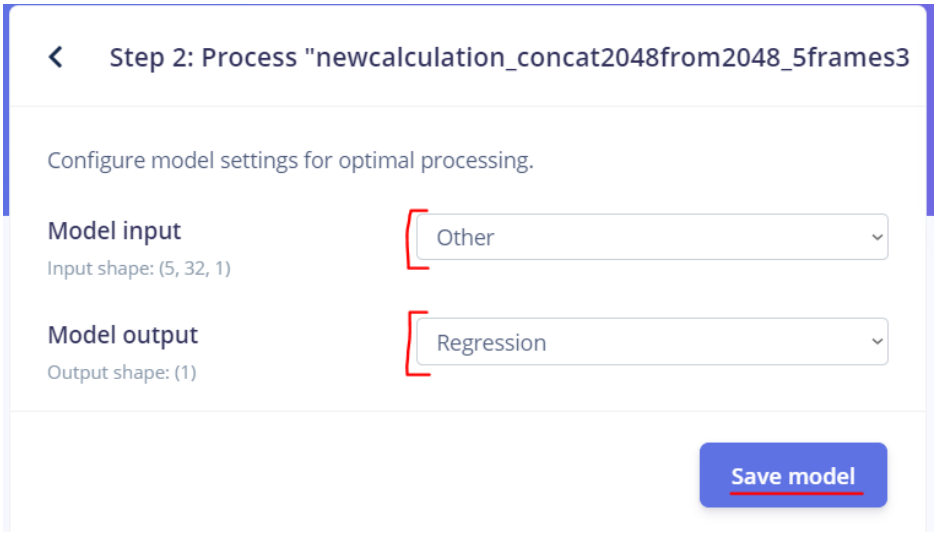
5.2.1.Après avoir créé un compte Edge Impulse et avoir créé un projet, aller dans “Dashboard” puis dans “Upload your model” :



5.2.2.Choisir le modèle au bon format et cliquer sur “Upload file” :

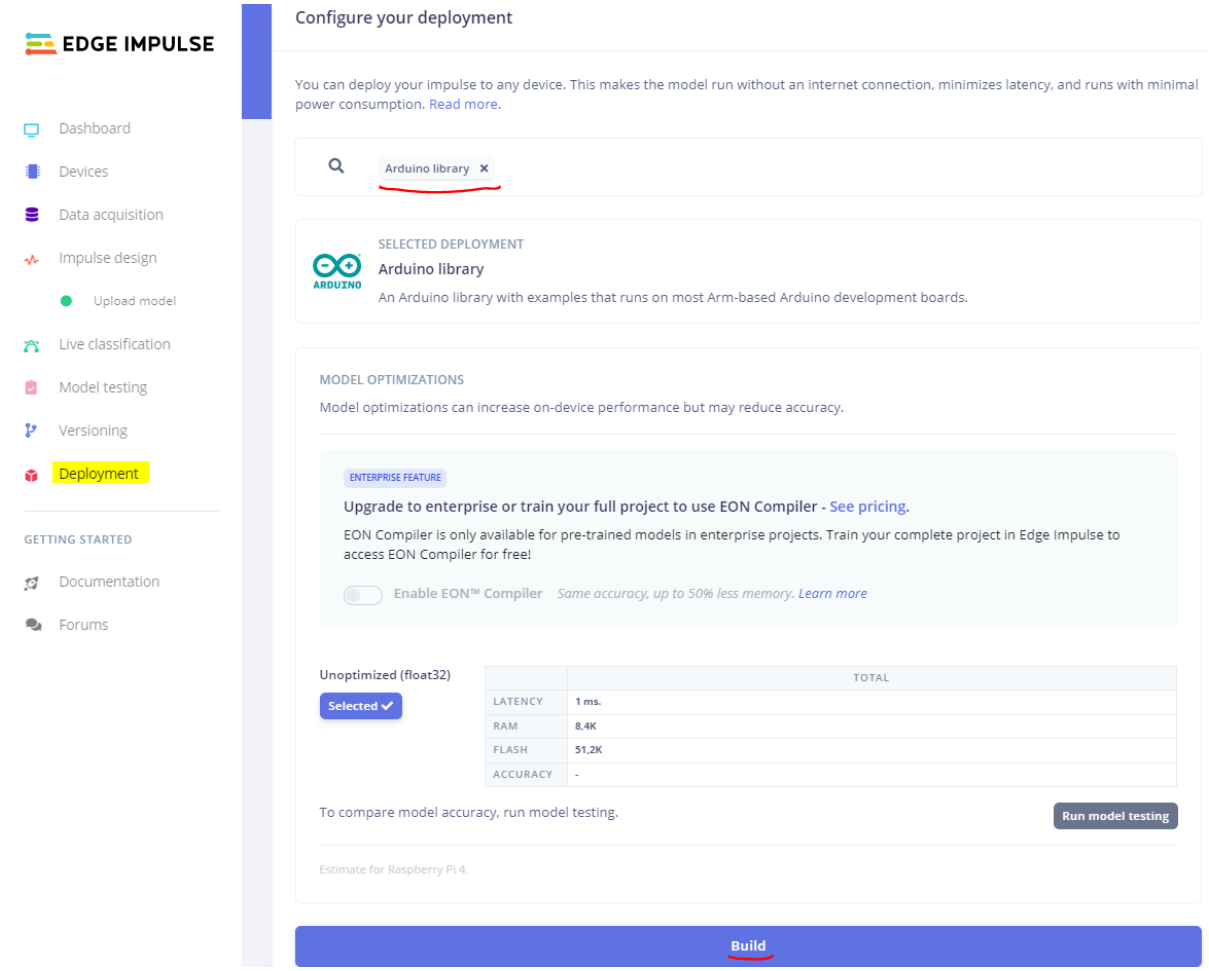


5.2.3.Choisir le format d'entrées et sorties du modèle et sauvegarder le modèle :

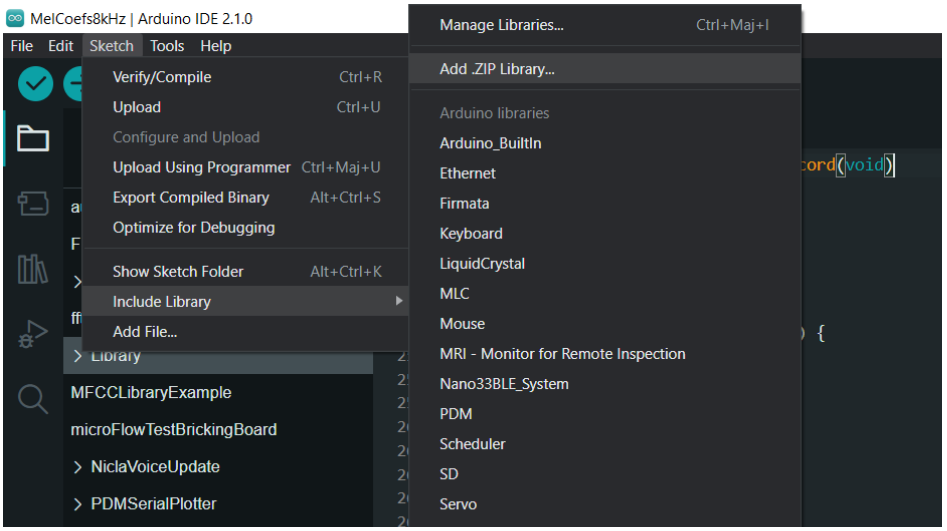


- Ici le format Other permet de correspondre aux entrées du modèle CNN, et la prédiction de sortie Regression est obligatoire lorsque la sortie est unique

5.2.4.Aller dans l'onglet "Deployment", choisir l'option de déploiement "Arduino library" et cliquer sur "Build" :



5.3.Importer la library Arduino générée avec l'IDE Arduino (Sketch → Include Library → Add .ZIP Library) :



6. Utilisation de la library Edge Impulse

6.1.Inclure la library importée :

```
#include <mbmn974-project-1_inferencing.h>
```

6.2.Fonction utilisée par la library Edge Impulse pour copier les features :

```
/**
 * @brief      Copy raw feature data in out_ptr
 *             Function called by inference library
 *
 * @param[in]  offset   The offset
 * @param[in]  length   The length
 * @param      out_ptr  The out pointer
 *
 * @return     0
 */
int raw_feature_get_data(size_t offset, size_t length, float *out_ptr)
{
    memcpy(out_ptr, coefficients + offset, length * sizeof(float));
    return 0;
}
```

6.3.Utilisation du modèle :

```
ei_impulse_result_t result = { 0 };

    // the features are stored into flash, and we don't want to load
everything into RAM
    signal_t features_signal;
    features_signal.total_length = sizeof(coefficients) /
sizeof(coefficients[0][0]);
    features_signal.get_data = &raw_feature_get_data;
```

```
    //Prédiction à partir des features traitées en mémoire flash et résultat stocké dans result
    EI_IMPULSE_ERROR res = run_classifier(&features_signal, &result, false);//  debug);
    if (res != EI_IMPULSE_OK) {
        //ei_printf("ERR: Failed to run classifier (%d)\n", res);
        return;
    }

    //Accès à la valeur prédite
    float value = result.classification[0].value;
```

6.bis.Exemples d'utilisation de la library disponibles dans File → Examples → "nom de la library" :