

Rapport Projet




06/04/2023
IA et Système embarqué


Antoine Binet
Léo Michiellin



Introduction



Ce projet nous a été proposé dans le but de développer nos compétences en intégration de réseaux neuronaux dans des dispositifs embarqués. Nous avons donc pu découvrir de nouvelles méthodes pour réduire la taille et augmenter la vitesse des réseaux de neurones, des méthodes telles que la quantification ou l'élagage (pruning).



Dans un premier temps, nous verrons comment nous avons pu mettre en place ces méthodes avec l'aide d'un document explicatif sur Deepnote. Ensuite, nous verrons comment implémenter le tout sur un dispositif embarqué (Arduino).

Partie I : Entraîner un modèle simple et le compresser avec tensorflow lite

Ce TP sur l'introduction à la quantification a été très instructif pour nous permettre de comprendre les différentes techniques visant à réduire la taille et augmenter la vitesse d'exécution des réseaux de neurones. Nous avons appris qu'il existait deux types de quantification : post-entraînement et entraînement conscient de la quantification, qui permettent de quantifier les poids et les activations des modèles en 8 bits pour réduire leur taille sans trop impacter leur précision. Nous avons également appris à utiliser la bibliothèque TensorFlow Lite pour convertir des modèles Keras en modèles TensorFlow Lite, ainsi qu'à spécifier des optimisations pour le convertisseur. Enfin, nous avons appris à créer un modèle avec entraînement conscient de la quantification en simulant les effets de la quantification pendant l'entraînement afin d'ajuster les poids du modèle en conséquence. Au final, nous avons compris que la quantification est une technique utile pour déployer des modèles sur des dispositifs embarqués disposant de ressources limitées, sans compromettre leur précision.

Vous pouvez trouver les résultats sur le Dossier du Github sous le nom de « rendu_Deepnote ».

Partie II : Transfert vers arduino

Dans cette partie, nous avons dû faire fonctionner notre modèle sur une carte Arduino. Pour cela, nous avons d'abord dû tester la carte avec un code simple permettant de vérifier si celle-ci fonctionne :

```
void setup() {
  pinMode(13, OUTPUT); // configure la broche 13 en sortie
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Ce code fonctionne, la LED clignote ! Maintenant il faut tester la caméra OV7670, pour cela c'est assez simple, on trouve un code d'exemple sur le logiciel Arduino IDE

```
#include <Arduino_OV767X.h>

unsigned short pixels[176 * 144]; // QCIF: 176x144 X 2 bytes per pixel (RGB565)

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("OV767X Camera Capture");
  Serial.println();

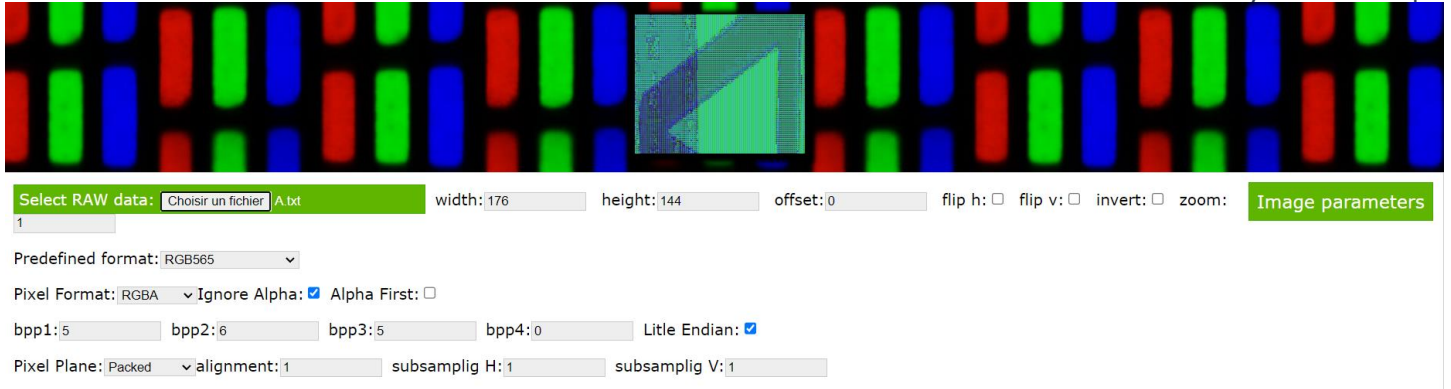
  if (!Camera.begin(QCIF, RGB565, 1)) {
    Serial.println("Failed to initialize camera!");
    while (1);
  }

  Serial.println("Camera settings:");
  Serial.print("\twidth = ");
  Serial.println(Camera.width());
  Serial.print("\theight = ");
  Serial.println(Camera.height());
  Serial.print("\tbits per pixel = ");
  Serial.println(Camera.bitsPerPixel());
  Serial.println();

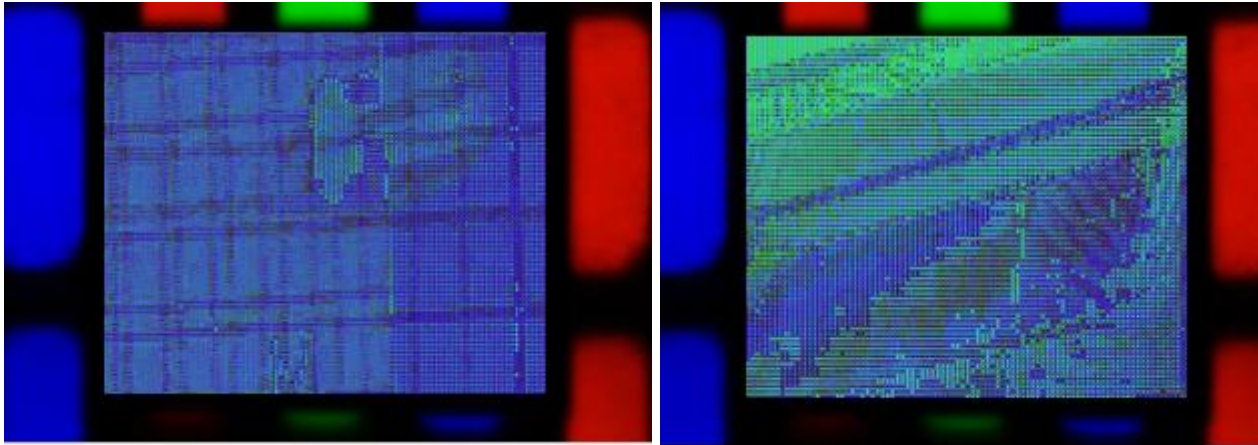
  Serial.println("Send the 'c' character to read a frame ...");
  Serial.println();
}
```

[illegible]

5



Test de photo de la lettre A



Test photo de la lettre M

Nous avons testé de nombreux paramètres différents et pris des centaines de photos sous divers angles, conditions lumineuses, etc. Mais nous n'avons pas réussi à obtenir de résultats concluants. Cela a été fatal pour la suite du projet, car nous avons dû finalement rendre la carte le 31/03, ni moi ni Léo ne pouvant rester sur Paris pour la rendre plus tard.

Nous avons donc dû finir le projet et écrire le code pour implémenter notre modèle sur la carte, sans la carte. Grâce à de nombreuses recherches sur Internet et des morceaux de code que nous savions fonctionnels, nous avons réussi à produire un code final, mais nous n'avons pas pu le tester. Nous savons donc qu'il n'est pas forcément fonctionnel ni optimal.

```
#include <Arduino.h>
#include <Arduino_OV767X.h>
#include <TensorFlowLite.h>

#include "C:\Users\binet\Desktop\IA\rednu project\detect_img\qat_model.h5" //modèle pré-
entraîné
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"

// constantes pour la cam
const int kNumCols = 176;
const int kNumRows = 144;
```

```

unsigned short pixels[kNumCols * kNumRows];

// constantes pour le modèle
const int kNumClasses = 10;
const int kTensorArenaSize = 4 * 1024;
uint8_t tensor_arena[kTensorArenaSize];

// variables pour l'inférence
float input_data[kNumCols * kNumRows];
float output_scores[kNumClasses];

void setup() {
    Serial.begin(9600);
    while (!Serial);

    // initialisation de la caméra
    if (!Camera.begin(QCIF, RGB565, 1)) {
        Serial.println("Erreur d'initialisation de la caméra !");
        while (1);
    }

    // initialisation du modèle
    const tflite::Model* model = tflite::GetModel(qat_model);
    tflite::MicroInterpreter interpreter(
        model, tflite::AllOpsResolver(), tensor_arena, kTensorArenaSize,
        &micro_error_reporter);
    interpreter.AllocateTensors();

    // config de TinyMLx
    TinyMLx.begin(&interpreter);
    TinyMLx.setOutputActivationParameters(0, 0.0, 1.0);

    Serial.println("Envoyez le caractère 'c' pour lire une image ...");
}

void loop() {
    if (Serial.read() == 'c') {
        Camera.readFrame(pixels);

        // on redim l'image pour fit le model
        ImageResize.resize(pixels, kNumCols, kNumRows, resized_image, kNewCols, kNewRows,
IMAGE_NEARESTNEIGHBOR);
        // On normalise la valeurs des pixels
        normalize_image(resized_image, input_tensor->data.f, kNumCols * kNumRows);

        // Exécution de l'inférence
        interpreter.Invoke();
    }
}

```

```

//on veut prendre le tensor en sortie et sa longueur
TfLiteTensor* output_tensor = interpreter->output(0);
const int output_length = output_tensor->bytes / sizeof(float);
//ces deux lignes on été trouvé sur internet (à vérifier si possible dans le futur)

// on cherche l'index avec la valeurs de confiance la plus grande
int best_index = 0;
float best_confidence = 0.0;
for (int i = 0; i < output_length; i++) {
    float confidence = output_tensor->data.f[i];
    if (confidence > best_confidence) {
        best_index = i;
        best_confidence = confidence;
    }
}

// affichage des predictions
Serial.print("Prediction: ");
Serial.print(best_index);
Serial.print(" (");
Serial.print(best_confidence, 4);
Serial.println(")");
}
}

```

Explication du code :

- 1- Importation des bibliothèques nécessaires
- 2- Définition de constantes pour les dimensions de l'image (kNumCols et kNumRows), le nombre de classes de l'inférence (kNumClasses) et la taille de l'espace de tenseur (kTensorArenaSize) alloué pour le modèle.
- 3- Initialisation de la caméra et du modèle. La fonction begin() est utilisée pour initialiser la caméra, et la fonction GetModel() est utilisée pour charger le modèle pré-entraîné dans la variable "model".
- 4- Boucle principale (dans la fonction loop()) qui attend la saisie du caractère "c" dans le moniteur série. Lorsque le caractère est reçu, l'image est capturée à partir de la caméra et stockée dans le tableau "pixels".
- 5- L'image est redimensionnée à la taille requise par le modèle et elle est normalisée
- 6- L'inférence est effectuée sur l'image normalisée à l'aide de la méthode "Invoke()" de l'interpréteur TensorFlow Lite.

- 7- Les résultats de l'inférence sont stockés dans le tenseur de sortie "output_tensor". La probabilité la plus élevée est extraite en recherchant l'index avec la valeur la plus élevée dans "output_tensor". Cette probabilité est stockée dans la variable "best_confidence", et l'indice correspondant est stocké dans la variable "best_index".
- 8- Les résultats de l'inférence sont affichés sous forme de prédiction de classe, avec l'indice de la classe ayant la probabilité la plus élevée et la probabilité elle-même.

Conclusion

Nous avons choisi de créer un binôme constitué d'Antoine Binet et de Léo Michiellin, car nous avons tous les deux une passion pour les systèmes embarqués et la possibilité de tester nos apprentissages sur de véritables cartes, telles que les Arduino ou autres microcontrôleurs. C'est pour cette raison que nous rendons aujourd'hui notre travail, avec le regret de ne pas avoir pu aller au bout. Nous espérons que la notation tiendra compte des travaux de recherche effectués ainsi que de l'implication que nous avons pu démontrer tout au long de ce semestre en communiquant régulièrement avec M. Cetinzoy.

Néanmoins, même si le projet n'a pas abouti à une solution fonctionnelle, nous avons tout de même pu développer nos compétences en implémentation de réseaux neuronaux. Ces compétences nous sont directement utiles pour nos stages de fin d'études, et nous souhaitons donc vous remercier pour cela.

PS : serait-il possible d'avoir une correction du code arduino pour implémenter le modèle afin de comparer avec notre solution ?