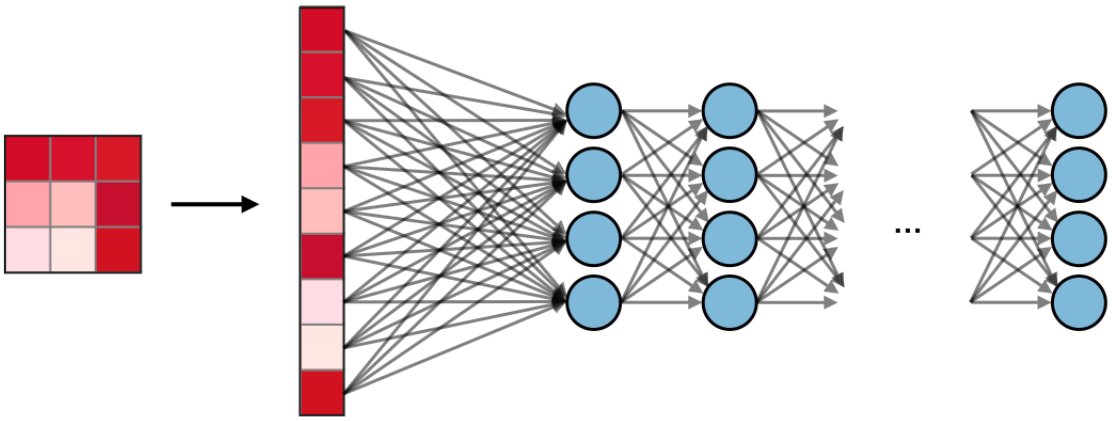
Université de Bretagne Occidentale | Faculté des Sciences & Techniques

Compte rendu de TAL n°3 Implémentation C de réseaux de neurones convolutifs sur systèmes embarqués

Ewann DELACRE

5 mai 2025

Professeur référent : Hai Nam Tran



1ère année de Master Informatique
Parcours Logiciels pour Systèmes Embarqués
Promotion 2025
Université de Bretagne Occidentale, Brest

Table des matières

1 Introduction

1.1 Du 28 avril au 2 mai

Ce troisième rapport présente les développements récents du projet, axés principalement sur l'intégration de bibliothèques d'inférence neuronale dans un environnement embarqué. L'objectif final reste l'exécution d'un réseau de neurones convolutif sur microcontrôleur, avec Zephyr RTOS tournant sur une Raspberry Pi Pico WH. Depuis le précédent rendu, les efforts se sont d'abord portés sur l'intégration de TensorFlow Lite for Microcontrollers (TFLite Micro) avec Zephyr. En raison de nombreuses difficultés techniques liées aux dépendances, un retour temporaire a été effectué vers une implémentation algorithmique manuelle d'un réseau de neurones en langage C. Ce document revient en détail sur ces étapes, les choix techniques associés, les problèmes rencontrés et les solutions mises en place.

2 Intégration initiale de TensorFlow Lite avec Zephyr

Nous avons commencé par chercher à intégrer TensorFlow Lite Micro dans notre environnement embarqué basé sur Zephyr. L'objectif était d'utiliser un modèle .tflite exporté depuis Python et d'en exécuter l'inférence sur la carte.

2.1 Configuration du projet

La configuration a nécessité :
- L'installation de Zephyr SDK et la configuration de west ;

- L'initialisation des modules requis (hal_stm32, cmsis, tensorflow, etc.) dans west.yml ;

- La compilation croisée via CMakeLists.txt en ajoutant les chemins vers TFLite Micro et ses dépendances.

2.2 Problèmes rencontrés

Des erreurs ont rapidement émergé :
- Problèmes de dépendances : conflits entre les versions de flatbuffers, absl, et les headers C++ de TFLite ;

- Structure de Zephyr rigide : les modules tiers comme TensorFlow nécessitent des ajustements précis pour être reconnus dans la hiérarchie de Zephyr ;

- Mémoire et complexité : la configuration de l'allocation mémoire (tensor_arena) et des opérateurs (AllOpsResolver) était difficile à stabiliser.

Face à ces obstacles, et dans une logique de développement itératif, nous avons décidé de mettre temporairement de côté l'intégration de TFLite pour nous concentrer sur une implémentation algorithmique contrôlée.

3 Implémentation simple en langage C

Pour reprendre la main sur le fonctionnement interne d'un réseau de neurones et s'affranchir des dépendances externes, nous avons rédigé un prototype en C (inference.c). Cette version vise à simuler toutes les étapes d'une inférence simple, à partir de zéro.

3.1 Structure du réseau

L'implémentation comprend :
- Une couche de convolution avec plusieurs noyaux 2x2 ;

- Une activation ReLU ;

- Une couche de Global Average Pooling ;

- Une fonction softmax ;

- Une prédiction par argmax.

Les données d'entrée sont une image 4x4, traitée par 3 filtres correspondant à 3 classes de sortie.

4 Implémentation via l'API C Tensor FLow

En parallèle, nous avons débuté l'intégration de TensorFlow C API (hors Zephyr) pour simplifier la création et l'exécution d'un graphe neuronal sans dépendre de l'environnement Zephyr. Le fichier tf_inference.c contient une première tentative d'exploitation d'un modèle TFLite, converti via xxd en tableau C, avec initialisation de l'interpréteur, allocation mémoire et inférence.
Ce code s'exécute pour l'instant en environnement desktop ou simulé, mais constitue une étape importante vers une future intégration embarquée.
Une fois encore, L'ensemble de ces implémentations est publié sur GitLab :

https://gitlab-depinfo.univ-brest.fr/e22306771/c-cnn

5 Bilan actuel et prochaines étapes

À court terme, les objectifs sont :

- Finaliser une inférence fonctionnelle en C avec TensorFlow, sur environnement PC puis embarqué ;

- Réessayer une intégration propre de TFLite dans Zephyr après stabilisation du projet ;

- Mesurer les performances et les ressources nécessaires selon les approches.