Bordeaux INP
ENSEIRB
MATMECA

Systèmes embarqués
PR 310
Rapport

L'intelligence artificielle sur un processeur Risc V

Élèves :
BELOUARGA Mohamed
EL ASFAR Abderrazak
OUAJIH Safae
EL MOUSSI Ghita

Enseignant :
KADIONIK Patrice

29 Janvier 2021

Bordeaux INP
ENSEIRB
MATMECA

Table des matières

1 Introduction

Dans le cadre du projet avancé cette année, notre mission principale était l'embarquement d'un réseau de neurones à convolution sur un processeur risc-V, afin qu'il puisse détecter les visages et les entourer via une caméra intégrée. Dans un premier temps, nous devrons développer un réseau de neurones et l'entraîner sur linux. Ensuite la finalité de la plus grande partie de notre travail visait l'adaptation du travail effectué sur linux avec la carte Sipeed Maixduino, tout en respectant ses contraintes matériels.

Le but de ce rapport est de vous exposer le contexte général dans lequel notre projet avancé s'est déroulé. Il présente dans un premier temps, la définition des termes cruciaux pour comprendre le cadre global du projet. En plus, il décrit soigneusement les démarches suivies, les recherches faites, les logiciels utilisés, et les tests effectués pour atteindre les résultats attendus.

2 Présentation de L'IA & Machine learning & Deep-Leanrning

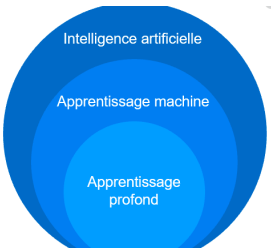
2.1 Intelligence Artificielle

L'intelligence artificielle (IA) est un ensemble de techniques permettant aux machines de simuler l'intelligence humaine, en accomplissant des tâches et en résolvant des problèmes normalement réservés aux humains. Les tâches relevant de l'IA sont souvent très triviales pour les humains, comme par reconnaître et localiser des objets dans une image.

Au début de l'apparition de cette discipline, construire un système intelligent consistait à écrire un programme « à la main », mais cette approche « manuelle » a ses limites. A titre d'exemple, pour une machine, une image est un tableau de nombres indiquant la luminosité (ou la couleur) de chaque pixel. Alors, comment une machine peut-elle identifier un chien dans une image quand l'apparence d'un chien peut varier infiniment ?

Quelques années plus tard, ce concept commence à associer l'intelligence aux compétences d'apprentissage. Autrement dit, une tâche peut augmenter ses performances avec l'expérience et l'apprentissage. Parce que, tout simplement, Il est virtuellement impossible de coder un programme non autonome et qui fonctionnera de manière efficace dans toutes les situations. C'est là qu'intervient l'apprentissage machine "Machine learning".

2.2 Le machine-learning



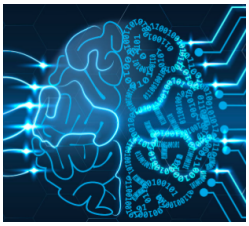
Machine learning est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux systèmes la capacité d'« apprendre » à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, il concerne la conception, l'analyse, l'optimisation, le développement et l'implémentation de telles méthodes.

Un système entraînable peut être vu comme une boite noire avec une entrée, par exemple une image et une sortie qui peut représenter la catégorie de l'objet dans l'image. On parle alors de systèmes de classification ou de reconnaissance des formes.

En Machine Learning classique, c'est le Data Scientist qui va lui même faire un choix, et extraire la donnée qui va influer sur la prédiction.Par contre, en apprentissage profond, c'est

l'algorithme qui va être entraîné pour sortir lui même les éléments influents dans la prédiction qu'on souhaite réaliser, C'est là qu'intervient le "Deep learning".

2.3 Deep Learning



Le Deep learning fait partie de la famille Machine learning. En effet, l'apprentissage profond utilise différentes couches d'unité de traitement non linéaire pour extraire et transformer les caractéristiques de la data ; chaque couche prend en entrée la sortie de la précédente ; les algorithmes peuvent être supervisés, non supervisés ou par renfoncement, et leurs applications comprennent la reconnaissance de modèles et la classification statistique.

D'ailleurs, les différentes architectures d'apprentissage profond telles que les réseaux de neurones profonds, les réseaux neuronaux convolutifs« convolutional deep neural networks » (L'architecture utilisé dans ce projet). elles ont plusieurs champs d'application, par exemple, la vision par ordinateur et reconnaissance automatique de la parole, et dans notre projet la détection de visage.

2.3.1 Les réseaux neuronaux convolutifs CNN

Les CNN représentent une sous-partie de réseaux de neurones et sont à ce jour un des modèles de classification d'images réputés être les plus efficace.
l'architecture du Convolutional Neural Network est composée d'une partie convolutive et une partie classification.

— Une partie convolutive : Son but ultime est d'extraire des caractéristiques propres à chaque image en les compressant de façon à restreindre leur taille initiale. En résumé, l'image crée en entrée passe à travers une succession de filtres, créant par la même occasion de nouvelles images appelées cartes de convolutions. à la fin, les cartes de convolutions obtenues sont concaténées dans un vecteur de caractéristiques nommé code CNN.

— Une partie classification : Le code CNN obtenu en sortie de la partie d'avant est l'entrée de la deuxième partie, qui est constituée de couches entièrement connectées appelées perceptron multicouche. L'objectif de cette partie est de combiner les caractéristiques du code CNN afin de classer l'image.

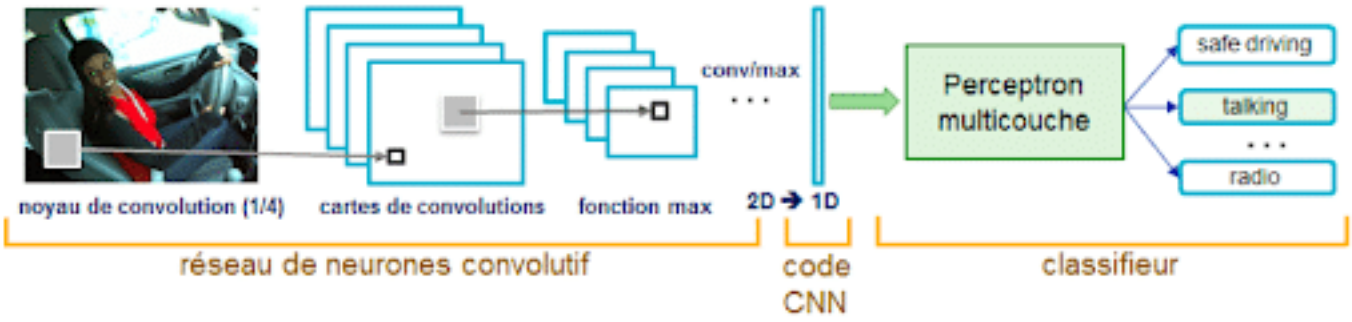


FIGURE 1 – Schéma représentant l'architecture d'un CNN

3 YOLO

YOLO (You Only Look Once) est un réseau de neurones de détection d'objets qui consiste à déterminer l'emplacement des objets sur l'image et les classifier. Sa grande force est la rapidité : il peut travailler en temps réel à 45 image/sec.

3.1 Vecteur de prédiction

Pour comprendre YOLO il faut comprendre son vecteur de prédiction. D'abord L'image d'entrée est divisée en SxS Blocs de pixels. Pour chaque objet présent sur l'image, un seul bloc de pixels est chargé de le prédire, c'est le bloc qui contient le point central de cet objet. Sur chaque bloc de pixels un nombre B de cadres prédits sont générés et C probabilités de classe qui est conditionné par les blocs de pixels contenant un objet : C = P(classe|objet), La prédiction des cadre se comporte de 4 composantes : (x, y, w, h). Les coordonnées (x, y) représentent le centre du cadre. Ces coordonnées sont normalisées pour qu'ils soient entre 0 et 1. Les dimensions du cadre (w, h) sont également normalisées à [0, 1], par rapport à la taille de l'image.Le réseau ne prédit qu'un seul ensemble de probabilités de classe par block, ce qui fait une probabilités de S*S*C au total.

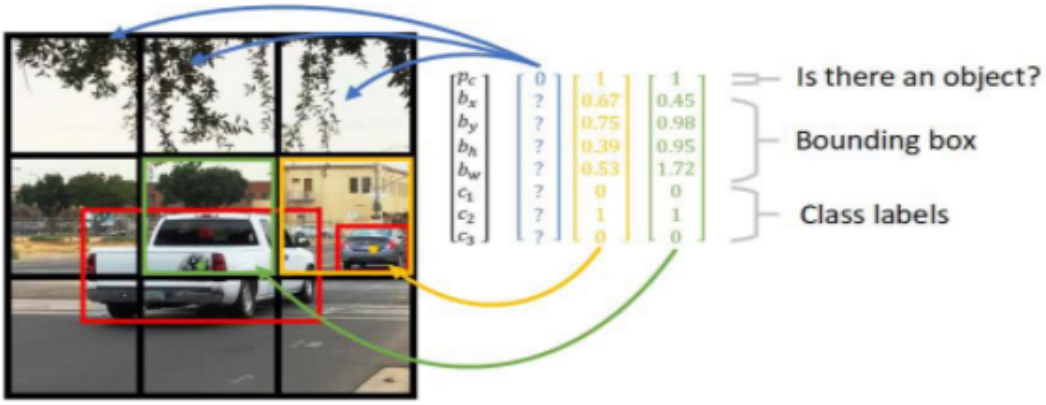


FIGURE 2 – Vecteur de prédiction

3.2 suppression non-max

Après l'étape de prédiction, nous aurons plusieurs cadres représentant le même objet, pour choisir un seul, nous appliquons le processus de suppression non-max qui se compose des trois étapes suivantes :

- Sélection du cadre qui a le plus élevé score en terme de probabilité.
- Calcule de son chevauchement avec toutes les autres cadres et suppression des cadres qui le chevauchent plus que le seuil choisi (0.5 par exemple)
- Répétition du processus jusqu'à la suppression de toutes les cadres qui ont un score inférieur à celui du cadre sélectionnée.

FIGURE 3 – Exemple d'application de la suppression non-max

4 MTCNN

MTCNN(Multi-task convolutional neural network) est un algorithme développé comme une solution pour la détection et l'alignement des visages. Le processus se compose de trois étapes de réseaux de neurones convolutifs capables de reconnaître les visages et les trais de visage tels que : les yeux, le nez et la bouche. la première étape de ce processus est la reconstruction de l'image qui consiste à dimensionner l'image avec différentes échelles et construire une pyramide d'images, pour détecter le visage avec différentes tailles.

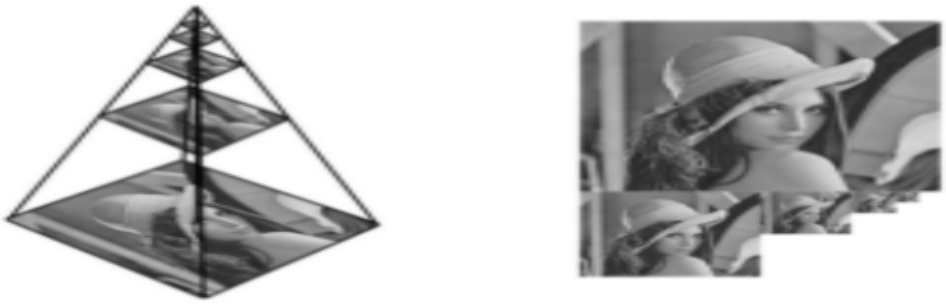


FIGURE 4 – Construction des images

4.1 Les trois étapes de MTCNN :

4.1.1 P-net

P-net(Proposal network) : est un réseau entièrement connecté (FCN), qui sert à proposer une zone de visage en suivant les trois étapes suivantes :

- Extraire les traits du visage initiaux
- Saisir ses traits dans trois couches de convolution de neurones pour identifier s'il s'agit d'un visage en utilisant un classificateur
- filtrer les cadres en utilisant une régression et le processus de suppression non-max

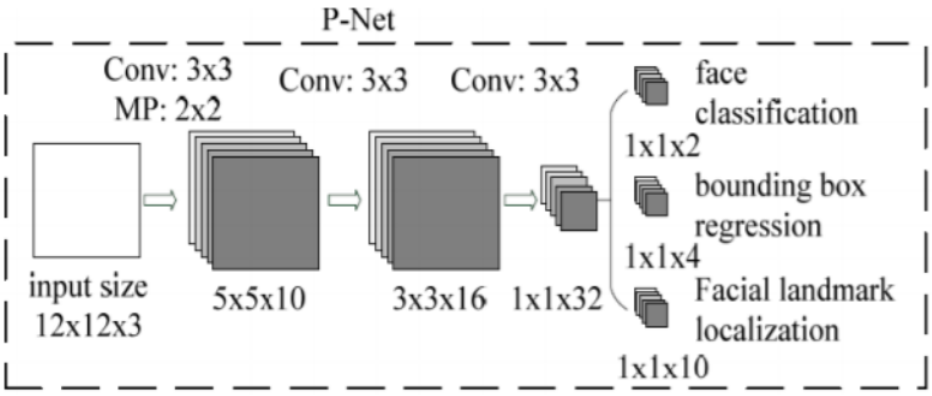


FIGURE 5 – Proposal network

A la sortie de ce premier network nous obtenons des propositions de zone de visage.

4.1.2 R-net

R-net (Refine Network) : consiste à filtrer plus de cadres prédits à partir des résultats de P-net avec une grande précision. R-net rajoute 128 réseaux entièrement connectés après la dernière couche de convolution pour enregistrer plus de caractéristiques d'image que P-net.

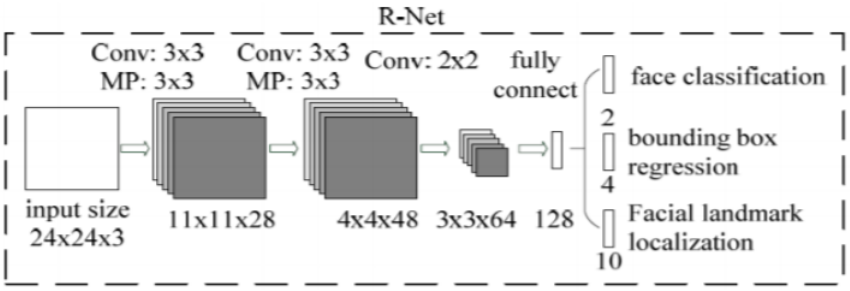


FIGURE 6 – Refine Network

4.1.3 O-net

O-net (Output Network) : Le réseau qui génère les cinq traits du visage en régressant les traits du visage.

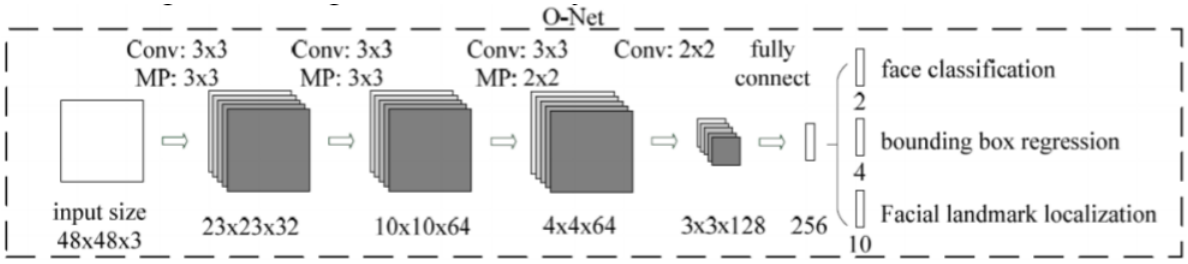


FIGURE 7 – Output Network

Après toutes les étapes précédentes, cet algorithme génère les coordonnées du coin supérieur gauche et les coordonnées du coin inférieur droit avec les cinq traits du visage.

5 Implémentation et Test

En se basant sur les deux networks précédentes nous avons créé et entraîné notre réseaux de neurones,et puis nous avons récupéré des images prises par la caméra de la carte Sipeed Maixduino pour faire le test suivant :

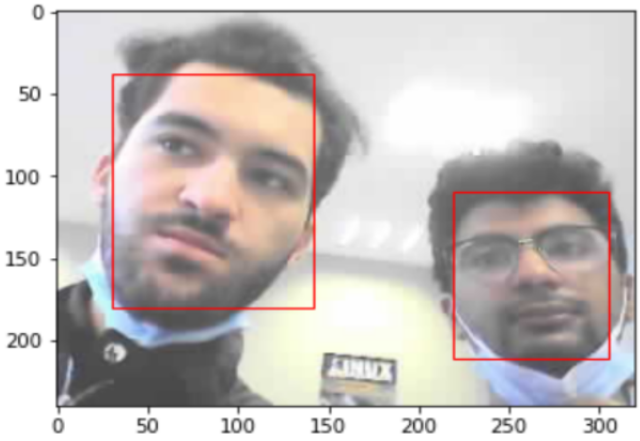


FIGURE 8 – Test du réseaux de neurones

Malheureusement nous n'avons pas pu embarqué ce réseau de neurones pour des raisons liées aux contraintes de la carte Sipeed Maixduino (la taille des fichiers à embarqué et des fonctionnalités qui ne sont pas supportés par kmodel : par exemple : PRelu).

6 Maixduino

La carte cible Sipeed Maixduino est une carte de développement RISC-V 64 au format Arduino Uno pour des applications d'intelligence artificielle IA et de l'internet des objets IoT. Cette carte fabriquée par SeeedStudio permet de réaliser des traitements à base d'intelligence artificielle directement au niveau de la carte, on peut faire de l'intelligence artificielle embarquée.

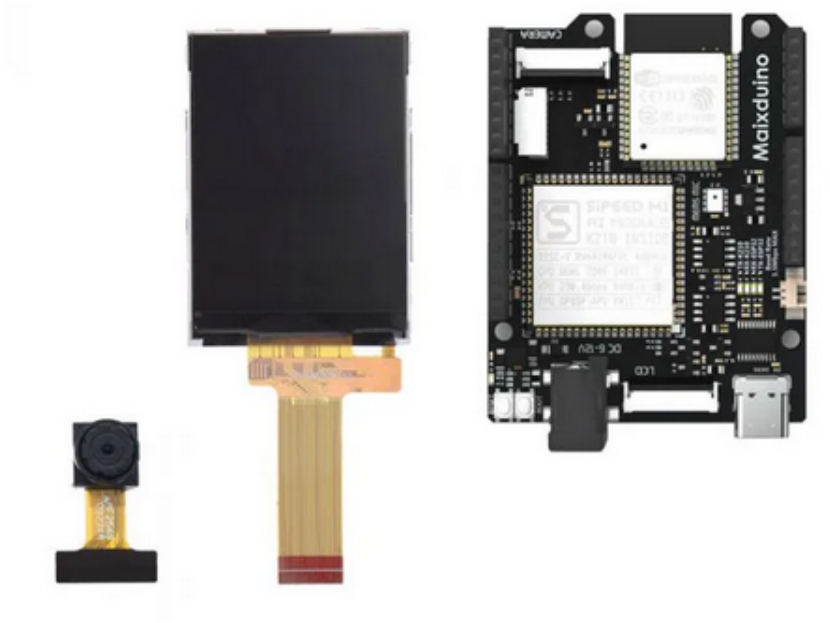


FIGURE 9 – Kit de développement IA Sipeed Maixduino

Spécifications techniques de la Sipeed Maixduino La carte cible possède les fonctionnalités suivantes :

- CPU : Module Sipeed M1
  - MAIX-IA IoT construit autour d'un processeur Kendryte KPU K210. Processeur RISC-V Dual Core 64 bits gravé à 28nm avec FPU@400 MHz (Processeur de réseau neuronal)
  - Accélération matérielle de reconnaissance d'image
  - 8MB de RAM
  - 16MB de mémoire flash interne
  - Support jusqu'à 8 microphones
  - Cryptographie AES SHA256
  - E/S I2C, SPI, I2S, WDT, TIMER, RTC, UART, GPIO
- Connectivité
  - Module ESP32 (ESP-WROOM-32) intégré
  - WiFi 2.4G 802.11. b/g/n
  - Bluetooth 4.2
- Interfaces
  - Connecteur GPIO femelle au pas de 2.54mm compatible avec les cartes d'extension Arduino Uno
  - Connecteur 24 broches pour caméra DVP QVGA@60FPS et VGA@30FPS
  - Connecteur 24 broches pour écran LCD 8-bits
- Audio

  - Microphone MEMS omnidirectionnel MSM261S4030H0 avec sortie I2S
  - Circuit audio DAC+PA
  - Connecteur pour haut parleur audio.
- Stockage
  - Lecteur de carte microSD
- Accessoires
  - Connecteur USB Type C pour alimentation, téléversement et debug
  - Convertisseur USB série CH552 offrant 2 canaux USB-TTL (2 ports série physiques)
  - Boutons RESET et BOOT
- Alimentation
  - Via connecteur USB-C
  - Via connecteur DC. 6 à 12V. Permet également d'alimenter des accessoires (servomoteurs, moteurs DC) 5V jusqu'à 1.2 A

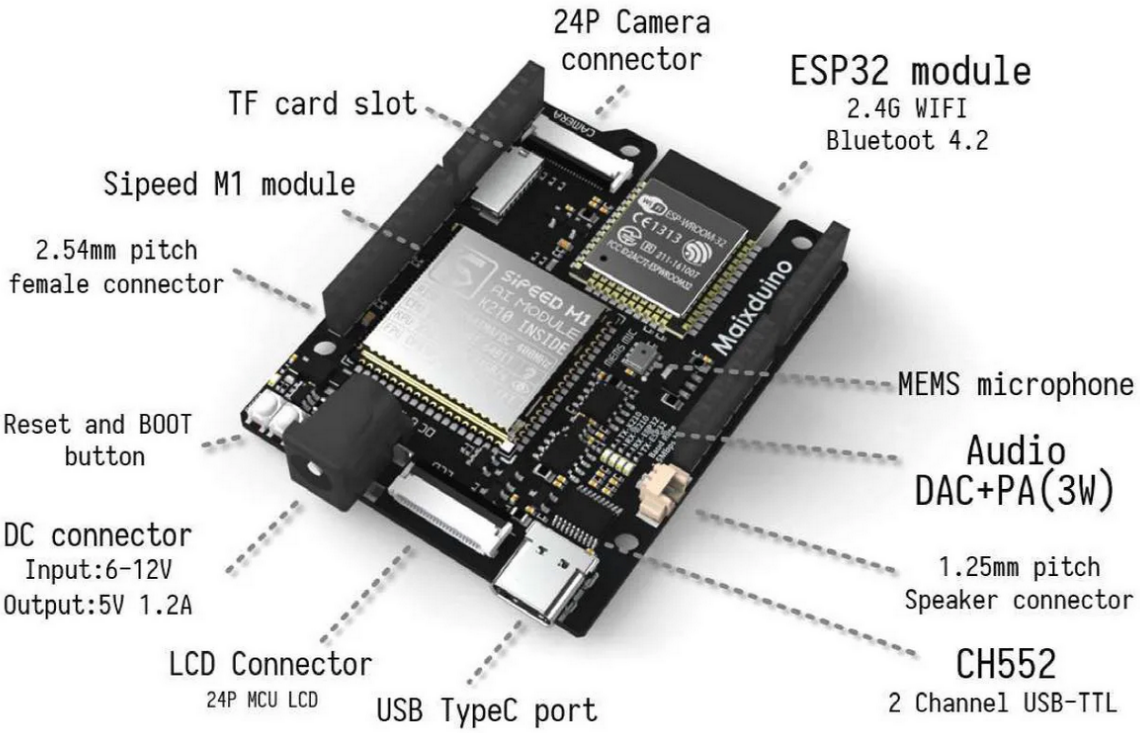


FIGURE 10 – Sipeed Maixduino

Maixduino est basée sur le module Maix AI de SeedStudio. La figure suivante montre une vue générale du module Maix AI.

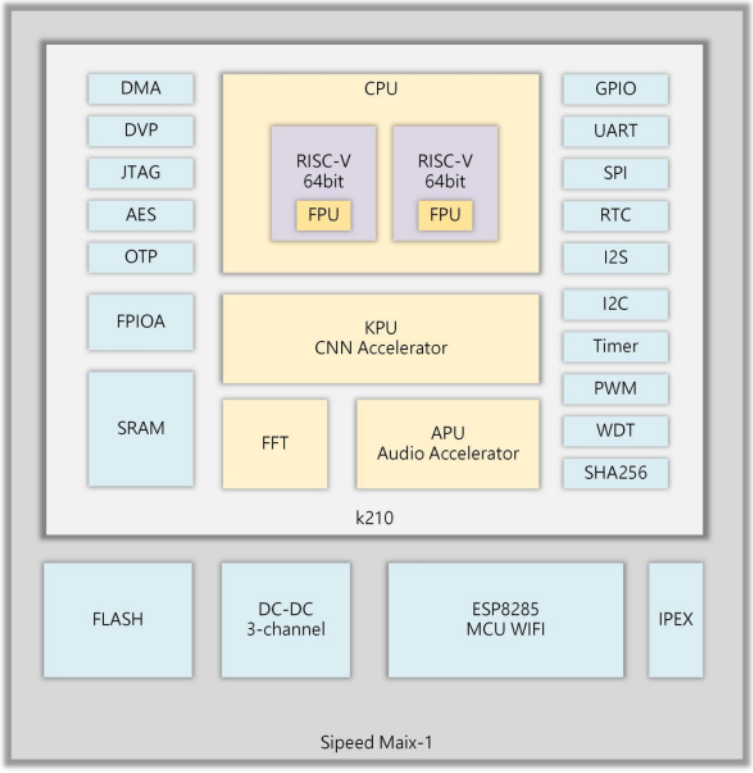


FIGURE 11 – Maix IA module

Pour faire de l'intelligence artificielle sur cette carte, SeeedStudio ont utilisé un KPU (Kernel processing unit), il s'agit d'une unité de traitement de noyau. Conçu pour accélérer le traitement des fonctions noyau individuelles sous-jacentes aux CNN, il fournit les implémentations matérielles de convolution, d'activation et de regroupement des fonctions noyau qui comprennent les différentes couches des modèles CNN.

Les caractéristiques du KPU

- Il supporte les modèles à virgule fixe.
- Supporte la configuration de chaque couche du CNN.
- La taille maximal du modèle peut être la taille de la mémoire flash moins la taille du logiciel.

La taille du modèle utilisé influence directement la détection temps réel des objets (Visages dans notre cas). En effet, pour des application dites temps réel, il faut utiliser des modèles à taille inférieure à 5.9 MB.

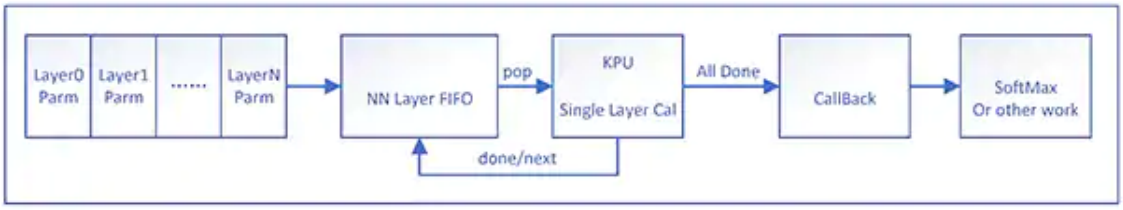


FIGURE 12 – La tâche KPU complète

La tâche KPU La figure ci-dessus représente une tâche complète du KPU. En fait, le KPU exécute les modéles de type fifo pour traiter de façon séquentielle chaque couche du CNN. Le

KPU lit les paramètres et les données de la première couche et exécute la fonction noyau de la couche en question. Un mécanisme de rappel permet aux développeurs d'exécuter leurs propres routines à mesure que le matériel KPU termine chaque séquence de traitement.

6.1 Microcode, Firmware

Les fichiers du microcode(Firmware) ont une extension .bin ou .kfpkg . Le firmware peut être téléchargé à partir du site officiel de maixpy :

https ://dl.sipeed.com/shareURL/MAIX/MaixPy/release/master

Nous avons utilisé pour notre application la version 4 du firmware. Il y a plusieurs choix , il faut noter que la version minimale ne supporte pas MaixPy IDE. Nous utilisons l'outil Kflash_gui pour implémenter le firmware. (Voir partie Kflash_gui)

6.2 Premiers tests

Seeedstudio a mis en place un modèle tiny yolo entraîné pour la détection de visages. En mettant le fichier .kmodel à l'adresse 0x300000 de la mémoire flash avec l'outil Kflash_gui et aprés exécution du script, nous avons validé le bon fonctionnement du modèle ainsi que notre bonne configuration de la carte. La figure suivante représente le résultat de la détection de visage avec le modèle fournit.

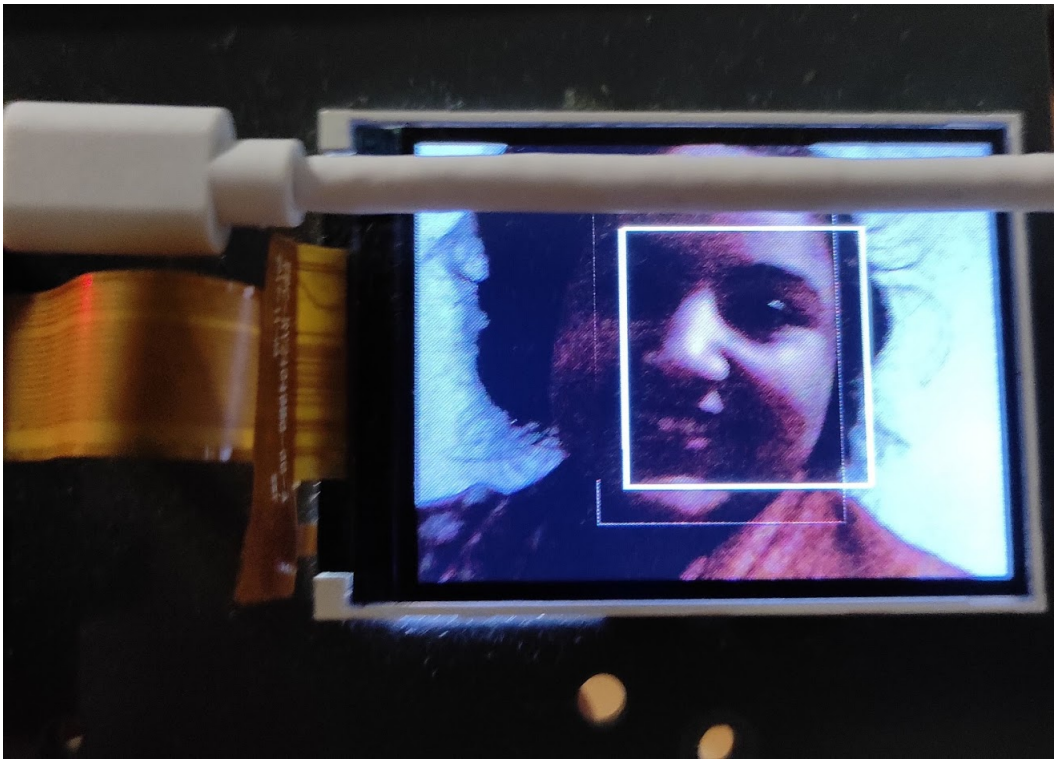


FIGURE 13 – Détection de visages : tiny-yolo

Nous avons aussi essayé d'autres modèles entraînés que nous avons trouvé au forum de Maixpy. Ci-dessous des captures des tests effectués pour un modèle de classification d'animaux. Les valeurs en rouge indiquent la probabilité de la classification.

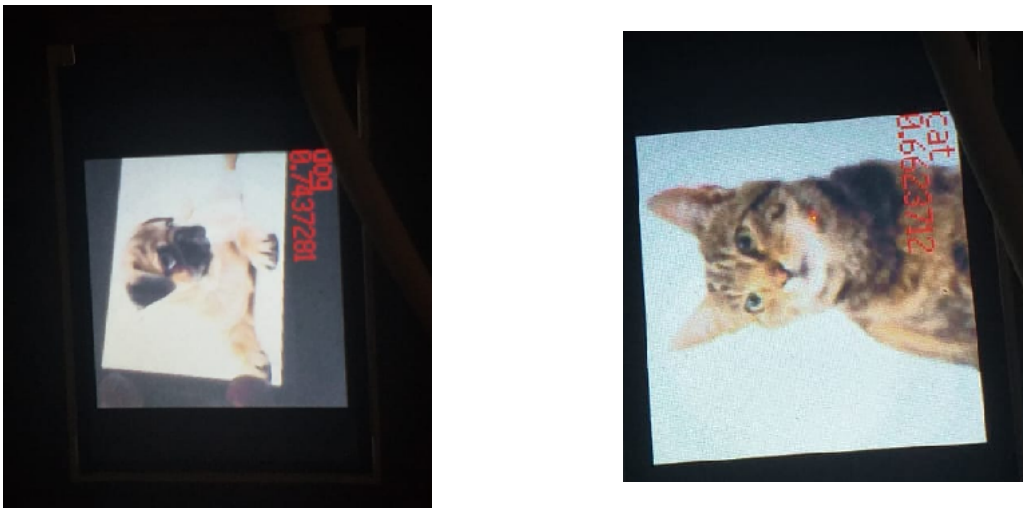


FIGURE 14 – Résultats d'un modèle de classification

Après avoir testé des modèles pré-entraînés sur la carte, nous avons essayé de faire notre propre modèle de détection de visages en passant par toutes les étapes d'entraînement du CNN et du validation. La partie (Axelerate) explique la démarche suivie.

7 Logiciels

7.1 Tensorflow et Keras

FIGURE 15 – Logo de Tensorflow

Tensorflow est une bibliothèque d'apprentissage automatique développée par Google, elle est devenue open source à partir de novembre 2015. Tensorflow est considérée comme la bibliothèque la plus utilisée dans le domaine du machine-learning, elle est supportée par plusieurs langages, le plus utilisé est Python. La communauté derrière Tensorflow est très vaste, contenant des entreprises, des chercheurs et des développeurs.

Tensorflow a annoncé, en Mai 2017, qu'elle va ajouter une nouvelle couche logicielle nommée Tensorflow Lite pour développer des programmes de l'IA légers, et spécifiques au développement des objets connectés et des applications Android, Tensorflow lite représente par exemple les int de 32 bits avec 16 bits ou 8 bits, ce qui réduit la taille d'un modèle, et donnent la possibilité aux microcontrôleurs d'exécuter tels modèles.

FIGURE 16 – Logo de Keras

Keras est une platforme haut niveau de l'IA basée sur Tensorflow, et qui rend la création des modèles, leur entraînement et la visualisation de leurs performances plus facile, nous pouvons dire que keras est bien 'user-friendly'.

7.2 Axelerate

(a) Logo de aXeleRate

(b) Dmitry MASLOV créateur d'aXeleRate

'aXeleRate' est un projet open source, qui simplifie l'embarquement de l'intelligence artificielle sur des cartes contenant MaixPy ; un projet portant l'environnement MicroPython sur des SoC K210. Ce projet, aXeleRate, était accompli par un ingénieur de Sibérie, et qui travaille à Shenzhen en chine, dans le domaine de l'IA embarquée. aXeleRate est un projet qui va nous permettre d'entraîner notre réseau de neurones basé sur un réseau classique qui s'appelle MobileNet alpha 0.75, ce modèle sera après transformé en fichier kmodel et exporté vers des puces k210.

En effet, aXeleRate transforme elle même le modèle entraîné .h5 en fichier .tflite, et le transforme encore en fichier .kmodel en utilisant un autre projet open source qui s'appelle nn_case. nn_case est spécialisé en transformation des modèles mais pour notre cas, nous allons l'utiliser pour faire des transformations tflite en kmodel.

Pour configurer aXeleRate, nous avons le choix entre l'utilisation d'un fichier de configuration et un dictionnaire(python) de configuration, pour notre cas, nous avons préférer d'utiliser un dictionnaire, pour fixer nos choix, notre modèle, notre base de données ...etc

7.3 Kflash_gui

Kflash_gui est un projet open source, qui utilise Python comme langage de script, ce programme est utilisé pour "formater" la carte, utiliser une autre version du firmware ou brûler un modèle sur la carte.

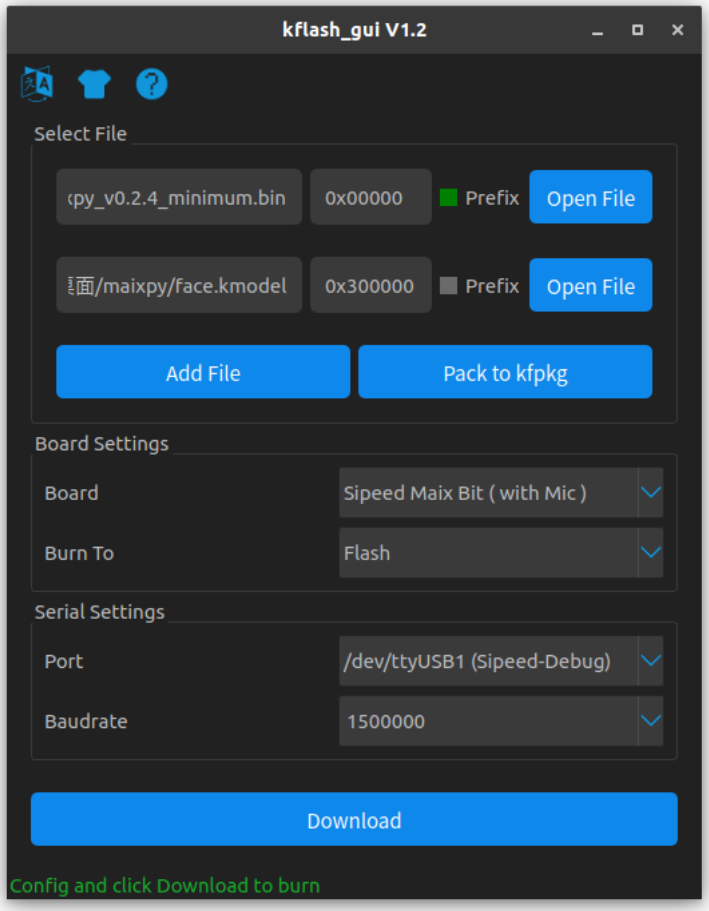


FIGURE 18 – Kflash_gui

7.4 MaixPy IDE

MaixpyIDE est un IDE qui utilise la syntaxe de MicroPython, cet IDE nous offre la possibilité de récupérer les images prises par la caméra de la carte utilisée, et en même temps voir les messages envoyés par la carte via le port série.

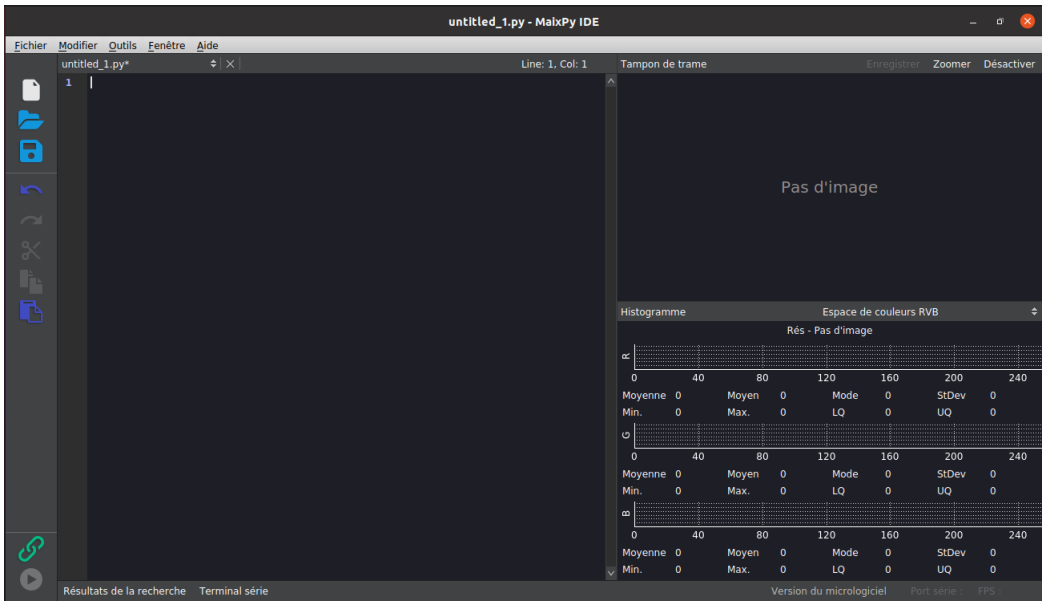


FIGURE 19 – MaixPy IDE

8 Procédure de réalisation du projet

Vous pouvez voir une démonstration de fonctionnement sur Colab via le lien suivant : https ://colab.research.google.com/drive/1E7wnVz1ZLMjMYIukVECIq5xDEfuLrI00 ?usp=sharing

8.1 Modèle

Model : "yolo"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv1_pad (ZeroPadding2D)	(None, 226, 226, 3)	0
conv1 (Conv2D)	(None, 112, 112, 24)	648
conv1_bn (BatchNormalization	(None, 112, 112, 24)	96
conv1_relu (ReLU)	(None, 112, 112, 24)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 24)	216
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 24)	96
conv_dw_1_relu (ReLU)	(None, 112, 112, 24)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 48)	1152
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 48)	192
conv_pw_1_relu (ReLU)	(None, 112, 112, 48)	0
conv_pad_2 (ZeroPadding2D)	(None, 114, 114, 48)	0

conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 48)	432
conv_dw_2_bn (BatchNormaliza	(None, 56, 56, 48)	192
conv_dw_2_relu (ReLU)	(None, 56, 56, 48)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 96)	4608
conv_pw_2_bn (BatchNormaliza	(None, 56, 56, 96)	384
conv_pw_2_relu (ReLU)	(None, 56, 56, 96)	0
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 96)	864
conv_dw_3_bn (BatchNormaliza	(None, 56, 56, 96)	384
conv_dw_3_relu (ReLU)	(None, 56, 56, 96)	0
conv_pw_3 (Conv2D)	(None, 56, 56, 96)	9216
conv_pw_3_bn (BatchNormaliza	(None, 56, 56, 96)	384
conv_pw_3_relu (ReLU)	(None, 56, 56, 96)	0
conv_pad_4 (ZeroPadding2D)	(None, 58, 58, 96)	0
conv_dw_4 (DepthwiseConv2D)	(None, 28, 28, 96)	864
conv_dw_4_bn (BatchNormaliza	(None, 28, 28, 96)	384
conv_dw_4_relu (ReLU)	(None, 28, 28, 96)	0
conv_pw_4 (Conv2D)	(None, 28, 28, 192)	18432
conv_pw_4_bn (BatchNormaliza	(None, 28, 28, 192)	768
conv_pw_4_relu (ReLU)	(None, 28, 28, 192)	0
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 192)	1728
conv_dw_5_bn (BatchNormaliza	(None, 28, 28, 192)	768
conv_dw_5_relu (ReLU)	(None, 28, 28, 192)	0
conv_pw_5 (Conv2D)	(None, 28, 28, 192)	36864
conv_pw_5_bn (BatchNormaliza	(None, 28, 28, 192)	768
conv_pw_5_relu (ReLU)	(None, 28, 28, 192)	0
conv_pad_6 (ZeroPadding2D)	(None, 30, 30, 192)	0
conv_dw_6 (DepthwiseConv2D)	(None, 14, 14, 192)	1728
conv_dw_6_bn (BatchNormaliza	(None, 14, 14, 192)	768

conv_dw_6_relu (ReLU) (None, 14, 14, 192) 0

conv_pw_6 (Conv2D) (None, 14, 14, 384) 73728

conv_pw_6_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_pw_6_relu (ReLU) (None, 14, 14, 384) 0

conv_dw_7 (DepthwiseConv2D) (None, 14, 14, 384) 3456

conv_dw_7_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_dw_7_relu (ReLU) (None, 14, 14, 384) 0

conv_pw_7 (Conv2D) (None, 14, 14, 384) 147456

conv_pw_7_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_pw_7_relu (ReLU) (None, 14, 14, 384) 0

conv_dw_8 (DepthwiseConv2D) (None, 14, 14, 384) 3456

conv_dw_8_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_dw_8_relu (ReLU) (None, 14, 14, 384) 0

conv_pw_8 (Conv2D) (None, 14, 14, 384) 147456

conv_pw_8_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_pw_8_relu (ReLU) (None, 14, 14, 384) 0

conv_dw_9 (DepthwiseConv2D) (None, 14, 14, 384) 3456

conv_dw_9_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_dw_9_relu (ReLU) (None, 14, 14, 384) 0

conv_pw_9 (Conv2D) (None, 14, 14, 384) 147456

conv_pw_9_bn (BatchNormaliza (None, 14, 14, 384) 1536

conv_pw_9_relu (ReLU) (None, 14, 14, 384) 0

conv_dw_10 (DepthwiseConv2D) (None, 14, 14, 384) 3456

conv_dw_10_bn (BatchNormaliz (None, 14, 14, 384) 1536

conv_dw_10_relu (ReLU) (None, 14, 14, 384) 0

conv_pw_10 (Conv2D) (None, 14, 14, 384) 147456

conv_pw_10_bn (BatchNormaliz (None, 14, 14, 384) 1536

conv_pw_10_relu (ReLU) (None, 14, 14, 384) 0

conv_dw_11 (DepthwiseConv2D) (None, 14, 14, 384) 3456

```
_________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, 14, 14, 384)       1536
_________________________________________________________________
conv_dw_11_relu (ReLU)       (None, 14, 14, 384)       0
_________________________________________________________________
conv_pw_11 (Conv2D)          (None, 14, 14, 384)     147456
_________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, 14, 14, 384)       1536
_________________________________________________________________
conv_pw_11_relu (ReLU)       (None, 14, 14, 384)       0
_________________________________________________________________
conv_pad_12 (ZeroPadding2D) (None, 16, 16, 384)        0
_________________________________________________________________
conv_dw_12 (DepthwiseConv2D) (None, 7, 7, 384)         3456
_________________________________________________________________
conv_dw_12_bn (BatchNormaliz (None, 7, 7, 384)         1536
_________________________________________________________________
conv_dw_12_relu (ReLU)       (None, 7, 7, 384)         0
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 7, 7, 768)       294912
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 768)         3072
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 768)         0
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 768)         6912
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 768)         3072
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 768)         0
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 768)       589824
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 768)         3072
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 768)         0
_________________________________________________________________
detection_layer_30 (Conv2D)  (None, 7, 7, 30)          23070
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 5, 6)        0
=================================================================
Total params : 1,856,046
Trainable params : 1,839,630
Non-trainable params : 16,416
```

Avant d'entraîner ce modèle, aXeleRate fait un transfert d'apprentissage, elle prend les poids d'imagenet et les attribue à notre modèle.

Ce modèle sera entraîné ensuite en utilisant la base de données CelebA, qui contient 202k images, mais pour ne pas dépasser les quotas de Colab, nous avons travaillé avec seulement 47k images. Avant de travailler avec ces images, il faut les préparer parce que nous avons un fichier qui contient la position originale du cadre(x,y) la largeur et la longueur du cadre, or nous devons avoir pour chaque image un fichier .xml contenant le nom de l'image, le label qui est évidemment un visage (a face) et les coordonnées du cadre(x_max,y_max,x_min,y_min). Le script que nous avons utilisé pour la préparation de la base de données est :

```
base_img = "/content/celebaForModel/"
base_xml = "/content/celebaForModelXML/"

def transform_x(x,b):
  return int((224.0/b)*x)

def transform_y(x,b):
  return int((224.0/b)*x)


def creat_xml(filename,xmin,xmax,ymin,ymax,TrainOrVal):
  imge = Image.open("/content/img_celeba_ancienne/"+filename+".jpg")
  imge_width = imge.size[0]
  imge_height = imge.size[1]
  wanted_width = 224
  wanted_height = 224
  newimge=imge.resize((wanted_width,wanted_height))
  newimge.save(base_img+TrainOrVal+"/"+filename+".jpg")
  a = open(base_xml+TrainOrVal+"/"+filename + ".xml","w+")
  a.write("<annotation>\n")
  a.write("<folder>celebAunziped</folder>\n")
  a.write("<filename>"+str(filename)+".jpg</filename>\n")
  a.write("<source>\n")
  a.write("<database>CelebA Database</database>\n")
  a.write("<annotation>CelebA</annotation>\n")
  a.write("<image>flickr</image>\n")
  a.write("</source>\n")
  a.write("<size>\n")
  a.write("<width>224</width>\n")
  a.write("<height>224</height>\n")
  a.write("<depth>3</depth>\n")
  a.write("</size>\n")
  a.write("<segmented>1</segmented>\n")
  a.write("<object>\n")
  a.write("<name>face</name>\n")
  a.write("<pose>Unspecified</pose>\n")
  a.write("<truncated>1</truncated>\n")
  a.write("<difficult>0</difficult>\n")
  a.write("<bndbox>\n")
  a.write("<xmin>"+str(transform_x(xmin,imge_width))+"</xmin>\n")
  a.write("<ymin>"+str(transform_y(ymin,imge_height))+"</ymin>\n")
  a.write("<xmax>"+str(transform_x(xmax,imge_width))+"</xmax>\n")
  a.write("<ymax>"+str(transform_y(ymax,imge_height))+"</ymax>\n")
  a.write("</bndbox>\n")
  a.write("</object>\n")
  a.write("</annotation>\n")
  a.close()

print("Mission finished!")
```

```
bbox = open("/content/drive/MyDrive/CelebA/Anno/list_bbox_celeba.txt","r")
from PIL import UnidentifiedImageError
import errno
print("training dataset is being prepared")
j = -2
for line in bbox:
  j=j+1
  if j>=1:
    try:
      infos = line.split()
      widthbox = int(infos[3])
      heightbox = int(infos[4])
      xmin = int(infos[1])
      xmax = xmin + widthbox
      ymin = int(infos[2])
      ymax = ymin + heightbox
      if j < 10:
        creat_xml("00000"+str(j),xmin,xmax,ymin,ymax,"train")
      elif j>=10 and j<100:
        creat_xml("0000"+str(j),xmin,xmax,ymin,ymax,"train")
      elif j<1000 and j>=100:
        creat_xml("000"+str(j),xmin,xmax,ymin,ymax,"train")
      elif j<10000 and j>=1000:
        creat_xml("00"+str(j),xmin,xmax,ymin,ymax,"train")
      elif j<100000 and j>=10000:
        creat_xml("0"+str(j),xmin,xmax,ymin,ymax,"train")
      elif j<1000000 and j>=100000:
        creat_xml(str(j),xmin,xmax,ymin,ymax,"val")
    except (UnidentifiedImageError, IOError):
      pass
  if j==45000:
    break
bbox.close()
print("Training dataset has been successfully prepared")

print("Validation dataset is being prepared")
j = -2
bbox = open("/content/drive/MyDrive/CelebA/Anno/list_bbox_celeba.txt","r")
for line in bbox:
  j=j+1
  try:
    if j>=45001:
      infos = line.split()
      widthbox = int(infos[3])
      heightbox = int(infos[4])
      xmin = int(infos[1])
      xmax = xmin + widthbox
      ymin = int(infos[2])
      ymax = ymin + heightbox
      if j < 10:
```

```
        creat_xml("00000"+str(j),xmin,xmax,ymin,ymax,"val")
      elif j>=10 and j<100:
        creat_xml("0000"+str(j),xmin,xmax,ymin,ymax,"val")
      elif j<1000 and j>=100:
        creat_xml("000"+str(j),xmin,xmax,ymin,ymax,"val")
      elif j<10000 and j>=1000:
        creat_xml("00"+str(j),xmin,xmax,ymin,ymax,"val")
      elif j<100000 and j>=10000:
        creat_xml("0"+str(j),xmin,xmax,ymin,ymax,"val")
      elif j<1000000 and j>=100000:
        creat_xml(str(j),xmin,xmax,ymin,ymax,"val")
  except (UnidentifiedImageError, IOError):
    pass
  if j==47000:
    break
bbox.close()
print("Validation dataset has been successfully prepared")

print("Mission finished!")
```

Ce script est une simple transformation de données, il prend les données d'un fichier .txt, et transforme ces données en fichiers .xml, et redimensionne les images afin d'avoir une base de données légère.

À l'exception de cette transformation de données, la démonstration est exactement ce que nous avions faite pour générer le fichier .kmodel.

8.2 Résultats et performances

Les performances que nous avons sont représentées ci-dessous :

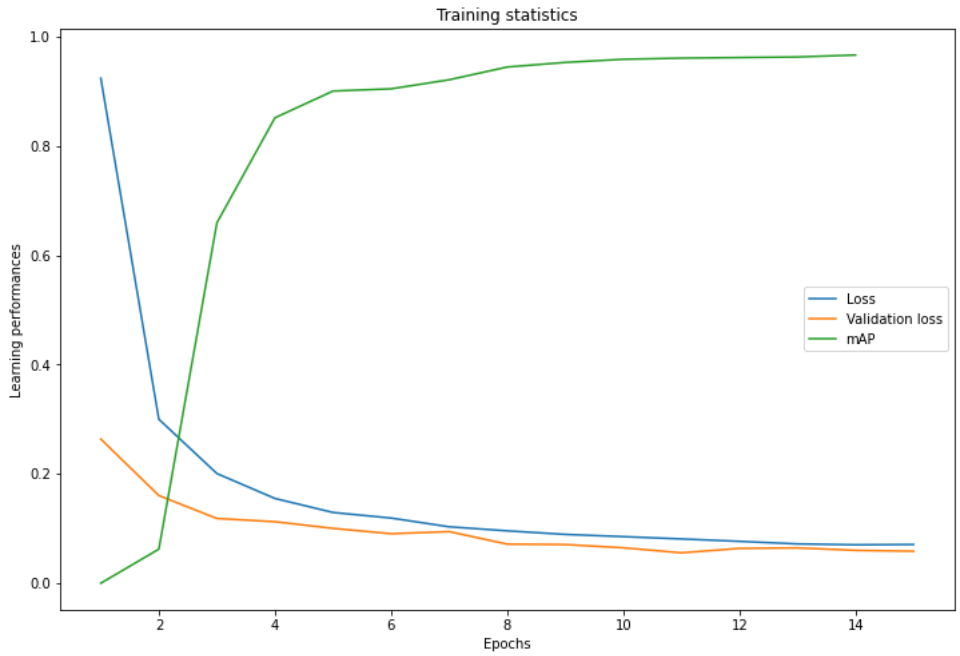


FIGURE 20 – Les performance d'entraînement

'mAP' (mean Average Precision) est la précision de notre modèle, mAP = $\frac{TP}{TP+FP}$, avec TP une prédiction correcte, et FP une prédiction fausse, cette variable donne une idée sur la précision de notre modèle.

'loss' est la fonction de perte, mais en fait, cette fonction donne l'erreur de prédiction, combien les cadres prédits sont précis, et si les cadres prédits sont vraiment des visages. 'val_loss' fait la même chose que la fonction 'loss' précédente, mais appliqué sur les images de validation, le modèle ne s'entraîne pas sur les images de validation, cela veut dire que nous ne lui donnons pas les positions des visages. Nous utilisons la base de validation pour savoir si le modèle a appris le concept d'un "visage" ou pas, si le modèle apprend bien à prédire les images d'entraînement et ne prédit pas les images de validation, cela veut dire que le modèle a appris à prédire seulement ce qu'il a déjà vu, ce phénomène est appelé 'Overfitting'.

Nous pouvons remarquer que le 'val_loss' était toujours inférieure au 'loss', alors, nous n'avons pas d'overfitting. Nous remarquons aussi que le modèle est très efficace car le mAP s'approche de 1 au dernier epoch.

8.3 Script de fonctionnement

```
1
2  import sensor
3  import image
4  import lcd
5  import KPU as kpu
6
7
8  lcd.init()
9  sensor.reset()
10
11 sensor.set_pixformat(sensor.RGB565)
12 sensor.set_framesize(sensor.QVGA)
13 sensor.set_windowing((224, 224))
14 sensor.run(1)
15 classes = ["face"]
16 task = kpu.load(0x300000)
17 a = kpu.set_outputs(task, 0, 7,7,30)
18 anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437, 6.92275,
       6.718375, 9.01025)
19 a = kpu.init_yolo2(task, 0.3, 0.3, 5, anchor)
20 while(True):
21     img = sensor.snapshot()
22     code = kpu.run_yolo2(task, img)
23     if code:
24         for i in code:
25             a = img.draw_rectangle(i.rect())
26     a = lcd.display(img)
27     print(code)
28 a = kpu.deinit(task)
```

9 Conclusion

Le but de ce projet était la découverte du monde de l'intelligence artificielle mais surtout de faire de l'IA embarquée. Nous avons choisi de travailler sur la détection de visages comme application. Le but ultime est donc d'embarquer un programme intelligent pour détecter les visages en temps réel. Nous avons commencé notre aventure dans le monde de l'IA par l'enrichissement de nos connaissances sur ce domaine, ces application, ces algorithmes mais aussi sur l'idée de migrer vers l'intelligence embarquée.

Dans un premier temps, nous nous sommes concentrés sur le développement des réseaux de neurones sur PC, nous avons réussi à tester plusieurs algorithmes (YOLO, MTCNN..). Nous avons créé, entraîné et aussi testé nos réseaux de neurones pour la détection des visages. Nous avons consacré la deuxième partie du temps à l'intelligence artificielle embarquée dans la fameuse Maixduino. Nous avons réussi à implanter un modèle fonctionnel d'un réseau de neurones entraîné par nous même pour la détection de visages. Le projet 'aXeleRate' nous a permis de comprendre les contraintes de l'IA embarquée. Nous avons appris tant sur l'IA que sur le matériel nécessaire pour l'implanter sur une cible. Nous avons réussi à obtenir des résultats très satisfaisants et la détection des visages par la Maixduino et avec notre modèle était un succès.

10 Ressources

- **Kendryte homepage :** https ://kendryte.com/
- **nncase converter :** https ://github.com/kendryte/nncase
- **Keras :** https ://keras.io/
- **tensorflow :** https ://www.tensorflow.org/
- **aXeleRate :** https ://github.com/AIWintermuteAI/aXeleRate.git
- **Kflash_gui :** https ://github.com/sipeed/kflash_gui
- **CelebA :** http ://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
- **MaixpyIDE :** https ://maixpy.sipeed.com/en/get_started/maixpyide.html
- **Maixduino :** https ://www.seeedstudio.com/Sipeed-Maixduino-for-RISC-V-AI-IoT-p-4046.html
  https ://maixpy.sipeed.com/en/
  https ://dl.sipeed.com/