Projet SY26
Détection de formes par réseau de neurones

Baptiste Bainier et Thomas Jeantet

21 décembre 2017

Introduction

Le développement de notre réseau de neurones a été implémenté dans le cadre de l'UV SY26. L'objectif est de pouvoir détecter et différencier des formes apparaissant sur un écran. Il y a 6 formes différentes, chacunes de couleur différente. Il faut pouvoir les différencier quelle que soit leur taille et leur position sur l'écran.

Le réseau doit être implémenté sur Raspberry Pi 3, et les images sont capturées par une Raspicam (v2).

Table des matières

1 Choix du framework

Dans un premier temps, nous avons du nous accorder sur le choix de l'environnement de travail. Deux options nous semblaient évidentes :

— Programmer en C++ avec OpenCV
— Programmer à l'aide d'un framework

Nous avons étudié les avantages et les inconvénients de chaque méthode avant de commencer la programmation.

1.1 Utiliser OpenCV

La première option évidente aurait été de dévlopper nous même tout le programme en C++ à l'aide des librairies OpenCV. Il existse quelques librairies OpenCV permettant d'implémenter des réseaux de neurones, mais ces librairies n'utilisent que des coiches MLP, ce qui est donc moins adapté que la framework que nous allons utiliser pour le traitement d'images.

De plus, nous avons estimé que développer tout le code nous même en C++ avec OpenCV nous aurait pris énormément de temps, ce qui, couplé aux technologies restreintes de OpenCV (MLP uniquement), en fait une méthode peu intéressante.

1.2 Utiliser un framework

La deuxième solution est donc d'utiliser un framework. Cette méthode présente également quelques inconvénients :

— Le temps de recherche des différents frameworks existants
— Le temps de formation au framework choisi

En effet, dans un premier temps nous avons dû nous rensigner sur les différents framework existants. Cette phase demande du temps, car il faut trouver tous les frameworks adaptés à notre sujet, puis trouver le plus adapté selonla documentation de chacun. Plusieurs frameworks se sont révélés intéressants, dont TensorFlow, PyTorch, ou Caffe que nous avons choisi.

Utiliser un framework présente aussi des avantages :

— Le temps de déploiement du réseau
— La flexibilité du réseau

Le réseau est plus rapide à mettre en place, car il existe déjà de nombreuses "briques" programmées, ce qui nous permet d'avoir à seulement les adapter à notre cas. On dispose également d'un réseau plus flexible, car tous les éléments sont prévus pour que les paramètre puissent être changés facilement, ce qui nous évite de refaire tout le code pour un moindre changement dans le premier scénario.

Nous avons donc choisi d'utiliser un framework, ce qui nous économise du temps de développement et nous permet de concentrer nos efforts sur l'optimisation du réseau de neurones plutôt que sur le développement de ce dernier. Nous avons choisi d'utiliser Caffe, car il semblait parfaitement adapté à notre projet (réseau de neurone pour la détection d'images), et de plus il est en grande partie rédigé en C++, qui est le langage que nous connaissons le mieux.

2 Prise en main de Caffe

Une fois le framework choisi, il faut le prendre en main. Il est nécessaire de comprendre son utilité et son fonctionnement avant de commencer à l'utiliser. Nous allons donc brièvement le présenter dans cette partie.

2.1 Du Caffe ?

"Caffe is a deep learning framework made with expression, speed, and modularity in mind." Caffe est un framework qui permet de créer des réseaux de neurones surmesure, selon les besoins de l'utilisateur. Plusieurs "couches" sont déjà implémentées, il suffit ensuite de choisir quelles couches utiliser, et quels paramètres leur appliquer. Cette étape de design du modèle se fait facilement au travers d'un simple fichier type protobuf, contenant l'ordre et les paramètres de chaque couche. De nombreux modèles différents existent déjà, et sont mis à disposition par Caffe au travers de leur "zoo".

L'utilisateur peut alors choisir d'utiliser une base de données existante, ou de créer sa propre database. De nombreux exemples sont déjà implémentés dans Caffe, ce qui nous permet de découvrir le fonctionnement du framework, mais aussi de nous inspirer de ces exemples pour créer notre réseau. Dans un premier temps, nous avons découvert le framework grâce à l'exemple du modèle LeNet, utilisé pour différencier les chiffres de la ma base de données MNIST.

2.2 L'exemple MNIST

Caffe fournit un exemple très complet basé sur la classification de la base de données MNIST à l'aide du modèle LeNet.

3 Adaptation du framework

3.1 Data augmentation

3.2 Création de database(s)

3.3 Personnalisation du net

4 Evaluation des performances

4.1 Influence du format des images

4.2 Influence des images de la database

4.3 Influence des hyperParamètres

Conclusion

4.4 Améliorer notre réseau

4.5 Le Machine Learning

4.6 Ce que le projet nous a apporté