DEEP

L'apprentissage profond pour l'indexation d'une grande base d'images

Stefan Duffner, Christophe Garcia

7 octobre 2019

1 Introduction

Dans ce projet des techniques d'apprentissage profond (Deep Learning) basées sur des réseaux de neurones à convolution (Convolutional Neural Networks, CNN) seront expliquées et mises en pratique. Ces algorithmes d'apprentissage automatique (Machine Learning) obtiennent actuellement les meilleurs résultats pour de nombreux problèmes dans le domaine de vision par ordinateur (Computer Vision) tels que la classification d'images et de vidéos, la détection d'objet, la segmentation d'image, la reconnaissance de visage et la reconnaissance d'activité, pourvu qu'il y ait un grand nombre d'exemples annotés d'apprentissage. Des systèmes, industriels et académiques, d'indexation d'images (par exemple de Google, Microsoft, Facebook) sont appris par ces méthodes avec des millions d'images annotées et capables de distinguer entre mille classes avec une grande précision.

2 Objectifs

Le but de ce projet est de mettre en œuvre un système de détection de visages par apprentissage profond, c'est-à-dire par réseau de neurones à convolution, en procédant en quatre étapes :

1. préparation et choix de données d'apprentissage,
2. choix de l'architecture du réseau de neurones,
3. implémentation du détecteur,
4. visualisation et évaluation des résultats

L'objectif est de comprendre :

— la difficulté d'extraire d'images des informations sémantiques,
— le fonctionnement d'un classifieur (basé CNN) et comment il est appris et appliqué,
— l'influence des données d'apprentissage, de l'architecture du réseau de neurones et le sur-apprentissage,
— le compromis entre précision et complexité d'un modèle appris,
— et enfin, les limites des méthodes d'apprentissage dans le contexte d'analyse d'images.

3 Organisation

Ce projet de conception est composé de 10 heures de travail encadré (TP) et 30 heures en autonomie. Le travail s'effectue en groupe de 4-5 personnes, est l'évaluation est basée sur un **rapport final à rendre avant le 15/11/2019**. Nous vous proposons de travailler sur une application de détection de visage.

4 Prérequis théoriques

Avant de démarrer, il est primordial d'avoir compris les concepts et fonctionnement des réseaux de neurones en générale, et des réseaux de neurones à convolution en particulier.

Voici quelques ressources sur ces sujets (en anglais) :

— Stanford – Deep Learning Tutorial : http://ufldl.stanford.edu/tutorial/
— Stanford – Introduction to CNNs : http://white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_Neural_Networks
— Tutoriel de Yann LeCun et Marc Ranzato sur le Deep Learning

D'une manière plus générale, le livre de Christopher Bishop donne une bonne introduction aux techniques d'apprentissage, comme les réseaux de neurones, et de reconnaissance de formes (« pattern recognition ») :

— Christopher M. Bishop. *Pattern recognition and machine learning.*, 2006, Springer

Un exemplaire est à la bibliothèque de l'INSA, et nous pouvons vous fournir une version PDF.

D'autres articles scientifiques fondamentaux et pertinents pour ce projet :

— Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 1998
— C. Garcia, M. Delakis. *Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection*. IEEE Transactions on Pattern Analysis & Machine Intelligence, vol.26, no. 11, pp. 1408-1423, 2004
— A. Krizhevsky, I. Sutskever, G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012

5 Prise en main de la bibliothèque

5.1 Introduction

Nous vous proposons d'utiliser la bibliothèque PyTorch en Python, très répandue dans la communauté Deep Learning. PyTorch est « open source » et basé sur Torch, une bibliothèque d'apprentissage machine très performante en Lua/LuaJIT.

Voici les principales ressources sur Internet :

— Site web : https://pytorch.org
— Documentation : https://pytorch.org/docs/stable/index.html
— Tutoriel :
  — Deep Learning with PyTorch (60 min.)
  — Learning PyTorch with Examples

5.2 Installation

Vous avez deux options d'utilisation : *avec* ou *sans* support GPU. Pour ce projet, vous n'avez pas besoin de GPU.

Pour l'installation de PyTorch, il y a plusieurs possibilité, et c'est relativement simple. Suivez les instructions sur le site web : https://pytorch.org.

5.3 Création d'un classifieur visage/non-visage

Pour commencer, nous vous fournissons un jeu de données d'apprentissage pour la détection de visages, contenant des exemples positifs (des imagettes de visages) et négatifs (des imagettes

coupées de manière aléatoire des photos sans visages). Toutes les images sont de taille 36×36 pixels et en niveaux de gris (format PGM).

En général, pour apprendre un réseaux de neurones, il faut trois ensembles de données :

1. ensemble d'apprentissage,
2. ensemble de validation,
3. ensemble de test.

L'apprentissage se fait uniquement sur l'ensemble d'apprentissage. On évalue la performance du modèle au cours de l'apprentissage sur l'ensemble de validation (par exemple toutes les N itérations), et on arrête l'apprentissage dès que l'erreur sur l'ensemble de validation commencer à augmenter (principe de « early stopping ») pour éviter le sur-apprentissage (overfitting). L'ensemble de test sert à évaluer la performance d'un modèle appris (donc après l'apprentissage !).

Nous vous fournissons deux ensembles : le « train » et le « test », dans les deux dossiers « train_images » et « test_images » respectivement avec plusieurs dizaines de milliers d'exemples. À vous de séparer les exemples « train » en « train » et « validation ». Pour une évaluation plus rigoureuse, la *cross-validation* est une technique courante. Affichez le contenu de ces dossiers pour regarder à quoi ressemblent les données d'apprentissage.

Charger et normaliser les images pour pouvoir être utilisé avec PyTorch, par exemple en utilisant *torchvision*. Le tutoriel de 60 minutes sur la base CIFAR10 : Deep Learning with PyTorch vous guidera pour le chargement, la normalisation, l'apprentissage et le test de votre classifieur.

6 Implémentation d'un détecteur

6.1 Procédé

Nous allons utiliser notre classifieur maintenant pour détecter des visages dans des images. Nous allons commencer par la méthode la plus simple, c'est-à-dire nous appliquons le CNN dans une (petite) fenêtre glissante sur une grande image pour déterminer la présence d'un visage ou non à chaque position. Ce processus est fait à plusieurs échelles. Donc, l'image est redimensionnée plusieurs fois (par un facteur de 1.2 par exemple) pour obtenir ce que l'on appelle une pyramide d'images. L'application du CNN (avec taille fixe d'entrée) sur cette pyramide permet de détecter les visages de différentes tailles. Enfin, les détections doivent être ramenées à l'échelle originale et être fusionnées. Ce traitement est loin d'être trivial et nécessite un certain nombre d'heuristiques (malheureusement). Regardez l'article de C. Garcia et M. Delakis pour en savoir plus.

Une optimisation consiste à supprimer le calcul redondant du CNN entre chaque fenêtre voisine. Pour cela, il faudrait remplacer les dernières couches du modèle (InnerProduct, avec des neurones) pour en faire des convolutions qui produisent des cartes en sortie. À vous de regarder comment l'implémenter en PyTorch.

6.2 Visualisation et évaluation

Pour vérifier que votre détecteur fonctionne, affichez les boîtes englobantes avec un « score » supérieur à un seuil sur quelques images que vous donnez à votre programme de détection. S'il y a beaucoup de fausses détections, c'est normal ! Il faut un classifieur avec un très bon taux de précision. À vous de l'améliorer en modifiant votre architecture de réseau de neurones, et la base d'apprentissage.

Pour l'évaluation quantitative, il y a des bases d'images (des photos) publiques avec des visages annotés (une boîte englobante pour chaque visage). En général, on rapporte le taux de bonnes détections (selon un critère de chevauchement) au fonction du nombre de fausses détections (que

l'on permet). Il y a un compris entre les deux, et selon le seuil sur le « score » du détecteur, on a plus de bonnes détection mais aussi plus de fausses détection ou inversement. On peut tracer une courbe en variant ce seuil, ce qu'on appelle la courbe ROC (Receiver Operator Characteristics).

Voici quelques bases publiques d'évaluation :

— CMU et CMU+MIT
— FDDB
— AFW

Il y a aussi un code récent d'évaluation sur trois bases (FDDB, AFW, et PASCAL faces) qui permet également de comparer son détecteur aux résultats de l'état de l'art. Il peut être téléchargé ici : http://markusmathias.bitbucket.org/2014_eccv_face_detection/. Nous pouvons vous aider à le mettre en place.

7 Améliorations

Nous vous proposons plusieurs pistes d'améliorations du code et des modèles que nous vous fournissons. Vous pouvez vous concentrer sur un ou plusieurs des points en dessous. Mais c'est à vous d'être créatifs !

Amélioration de la performance du classifieur : cela se fait en cherchant une architecture plus efficace et plus performant du CNN, et en augmentant la base d'apprentissage. Il y a beaucoup de ressources sur Internet avec des images de visages. Par exemple :

— https://facedetection.com/datasets
— http://www.face-rec.org/databases/
— http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html
— http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/
— http://lrs.icg.tugraz.at/research/aflw/

Nous disposons de certaines bases que nous pouvons vous fournir directement.

Améliorations algorithmiques : vous pouvez travailler sur l'optimisation du détecteur en terme de temps de calcul. La suppression de la redondance mentionné dans la section 6 est une première piste. Et un certain nombre de choses pourraient être parallélisées (avec l'utilisation de GPU et/ou plusieurs cœurs de CPU). Faites des mesures de temps et/ou du « profiling » pour comparer.

Extension multi-vues : vous pouvez apprendre un CNN qui estime non seulement la présence ou non d'un visage mais aussi son orientation. Par exemple, il peut y avoir six classes : non-visage, profil gauche, semi-profil gauche, frontal, semi-profil droit, profil droit. Ou deux modèles, le premier qui détecte le visage, et le deuxième son orientation. Il faut des images annotées, bien entendu.

Extension multi-modale : jusqu'ici nous avons seulement traité des images en niveaux de gris. Rien nous empêche d'utiliser les images en couleur. Il faut donc créer un ensemble d'apprentissage en couleur et modifier l'architecture du CNN pour prendre les 3 canaux en entrée. Une autre idée et d'utiliser des vidéos, donc des séquences d'images.

Extraction d'autres informations : l'estimation de la pose de la tête mentionné ci-dessus est une information qui peut être intéressante. Mais on peut aussi s'intéresser au genre, par exemple, aux caractéristiques du visage (présence de lunettes, barbes etc.), ou reconnaître des expressions faciales (par exemple le sourire). Enfin, on peut apprendre l'identité de la personne et ainsi créer un système de reconnaissance faciale avec un nombre limité de personnes (monde fermé).