TP Réseaux de Neurones pour l'image

Matthieu CORD Thomas ROBERT Arnaud DAPOGNY
Micael CARVALHO Rémi CADENE

Multimedia 2018

1 Introduction

Aujourd'hui, avec l'utilisation massive des smartphones et des réseaux sociaux, les images sont omniprésentes dans notre vie quotidienne. Pour traiter et exploiter cette masse de données, il est important d'avoir des systèmes de reconnaissance, pour analyser et interpréter le contenu visuel des images.

Depuis les 10 dernières années, le Deep Learning, un sous domaine du Machine Learning qui regroupe des modèles appelés Réseaux de Neurones Profonds, a beaucoup impacté la Vision Artificielle en permettant d'atteindre des performances inégalées sur de nombreuses tâches de vision comme la classification d'images, la localisation d'objets, la segmentation d'images, etc. Cette avancé technologique a notamment permis le développement ou/et l'amélioration de nombreux produits industriels (Google Photos, Flicker, Facebook Vision, Uber SmartCar, etc.).

Ces progrès ont été rendus possible grâce à l'augmentation de la puissance de calculs et du nombre de données annotées. Ces deux facteurs ont permis l'entrainement de Réseaux de Convolutions sur des bases d'images large échelle de plusieurs millions d'images. Ces réseaux peuvent être ensuite utilisés sur d'autres bases plus petites et d'autres tâches (transfer learning) comme extracteur de features remplaçant les systèmes "handcrafted" plus classiques tels que les SIFT.

2 Objectifs du TP

Nous allons nous intéresser au problème de la classification d'images, où l'objectif est de prédire si une catégorie sémantique (e.g. voiture) est présente dans l'image, à partir de son contenu visuel. Nous ne nous intéresserons pas à l'apprentissage de réseaux de convolutions, mais à l'exploitation de réseaux pré-entrainés comme extracteur de features.

L'objectif de ce TP est de retrouver les résultats et conclusions présentés dans Chatfield *et al.* (2014) [1].

Dans leur article, Chatfield *et al.* ont étudié les performances de features extraites grâce à différents réseaux de neurones profonds sur le dataset PASCAL VOC 2007.

Le TP s'organise comme suit :

- Etude du dataset
- Etude d'un réseau de convolution pré-entrainé
- Extraction de features du réseau
- Entrainement d'un modèle de classification
  - Recherche des hyperparamètres du modèle (train/val)
  - Evaluation du modèle (trainval/test)

Nous vous fournissons une grande partie du code (python). Il vous faudra le lire, le comprendre, et y apporter quelques modifications pour pouvoir faire l' etude correctement.

Vous devrez rendre un rapport final (pdf) par mail et une archive de votre code (zip) a la fin de l'intégralité des séances. Votre rapport en cours de rédaction devra néanmoins nous être envoyé par mail jusqu'au samedi suivant à chaque séance, 18h : *thomas.robert@lip6.fr* et *arnaud.dapogny@lip6.fr*. Ce rapport devra être organisé avec une introduction et une conclusion. Pour chaque partie, nous vous donnons une liste de questions qui permettra l'orientation de la rédaction de votre rapport. Vous serez évalués sur votre capacité à nous présenter les concepts nécessaires à la résolution de la tâche proposée dans ce TP (définir les termes, afficher des courbes, présenter des tableaux de résultats). Nous vous encourageons à aller plus loin (par exemple étudier différents réseaux).

3 Installation

Nous avons déjà installé toutes les bibliothèques nécessaires à Polytech. Si vous voulez travailler depuis chez vous, nous vous conseillons d'utiliser ssh et sshfs. Sinon vous devrez installer :

- Python 3.6 avec Conda : `https://conda.io/docs/install/quick.html`
- PyTorch : `http://pytorch.org`
- Sklearn : `http://scikit-learn.org/stable/install.html`

Dans les deux cas, ouvrez votre terminal et copiez les lignes suivantes :

```
pip3 install --user --upgrade numpy ipdb torchvision==0.1.6
git clone https://github.com/ThomasRobertFr/polytech-2018.git # telecharge le
cd polytech-2018/code # change de dossier
python3 visu.py # test le code de visualisation du convnet
python3 train.py --help # affiche les arguments en ligne de commande
python3 train.py --C 4 # entraine un classifier avec un C=4
```

Les deux fichiers principaux sont composés d'un certain nombre de fonctions (`def fonction(params)`) et d'un *main* qui est le code exécuté lorsque l'on exécute le ficher. Il est contenu dans le bloc `if __name__ == '__main__'`.

Pour arrêter l'exécution du code quelque part et pouvoir manipuler les objets (variables) existants à cet endroit, ajoutez `import ipdb; ipdb.set_trace()` dans le code là où vous voulez que l'exécution soit mise en pause.

4 Étude des datasets

4.1 ImageNet (ILSVRC compétition)

ImageNet est une base de données constituée de 1.4 millions d'images. Chaque image est associée à une classe parmi 1000.

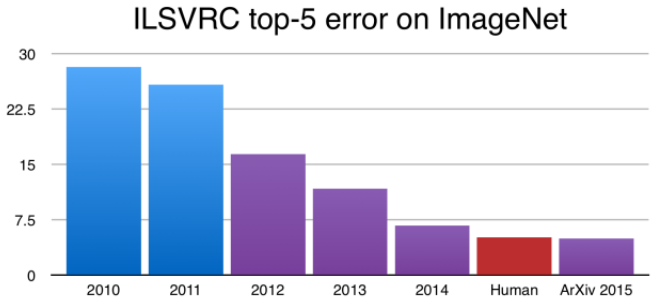

FIGURE 1 – L'erreur top-5 sur la base ImageNet en fonction des années (en bleu les modèles handcrafted, en violet les modèles incorporant des réseaux de convolutions).

Questions :

— Combien de classes ? (En citez quelques unes)
— Combien de classes par image ?
— Combien d'images en train, test ?
— Est-ce un dataset multi-labels ou multi-classes ?

4.2 Pascal VOC2007

PASCAL VOC 2007 est une base de données constituée de 9963 images multi-labels, stockées dans le dossier `/home/sasl/shared/EI-SE5-CS/datasets`. Nous considérons le problème comme une succession de problèmes binaires *one versus rest*. En d'autres termes, pour chaque classe c, chaque image I a le label $+1$ si elle contient l'objet c, et le label -1 sinon.

L'ensemble des images est divisé en trois groupes : *train*, *val* et *test*. Ces groupes déterminent le rôle de chaque image dans l'étude :

— les images de *train* et de *val* seront utilisées pour apprendre le SVM (recherche d'hyperparamètres) ;
— les images de *test* ne seront utilisées que pour évaluer le SVM ainsi appris (sur *trainval*)

Note : certaines images ont, pour une classe donnée, le label `0` : ce sont des images « difficiles ». *Pour cette étude, elles ne seront tout simplement pas prises en compte, ni en apprentissage, ni en test.*

Questions :

— Combien de classes ?
— Combien de classes par image ?
— Combien d'images en train, val, test ?
— Qu'est-ce que trainval ?
— Quelle est la différence entre val et test ?

5 Étude d'un réseau de convolution pré-entrainé

Un réseau de neurones y est représenté comme une structure à plusieurs couches de traitement. Plusieurs types de traitement sont implémentés, dont notamment ceux utilisés pour ce travail :

— couche de convolution
— couche de pooling
— couche de ReLU
— couche de softmax

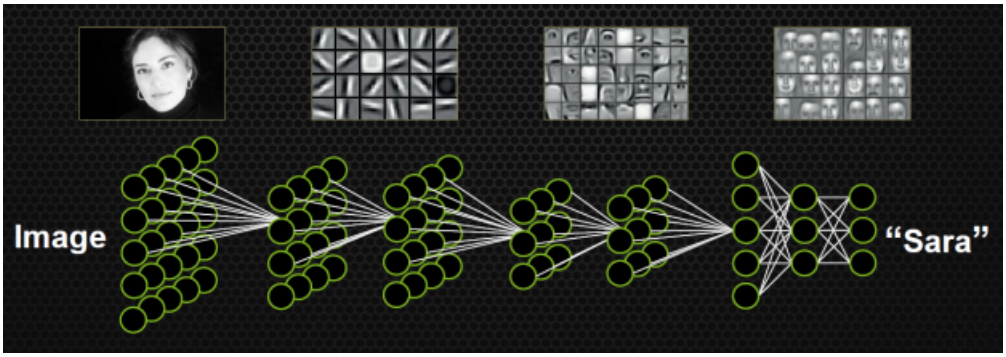

FIGURE 2 – Représentation intuitive d'un réseau de convolution pour la classification.

Il est courant d'appliquer un prétraitement aux entrées d'un réseau de neurones (dans notre cas des images). Il existe de nombreuses façon différentes de faire cette opération.

Dans le cas du papier Chatfield *et al.* (2014), les images doivent être redimentionnées (avec déformation si nécessaire) à une taille fixe (224×224 pixels). Il faut ensuite « centrer »les

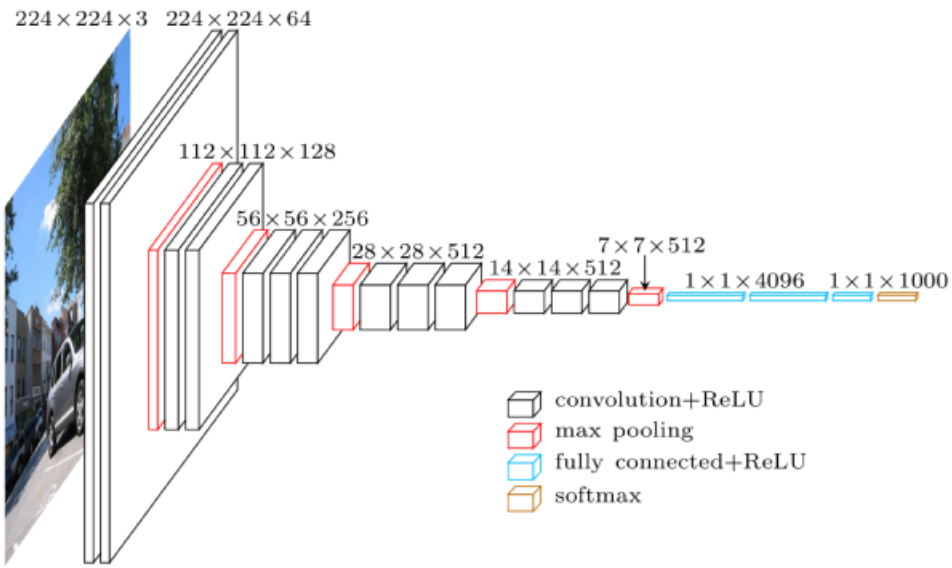

FIGURE 3 – Représentation détaillée des sorties de chaque couche de VGG16.

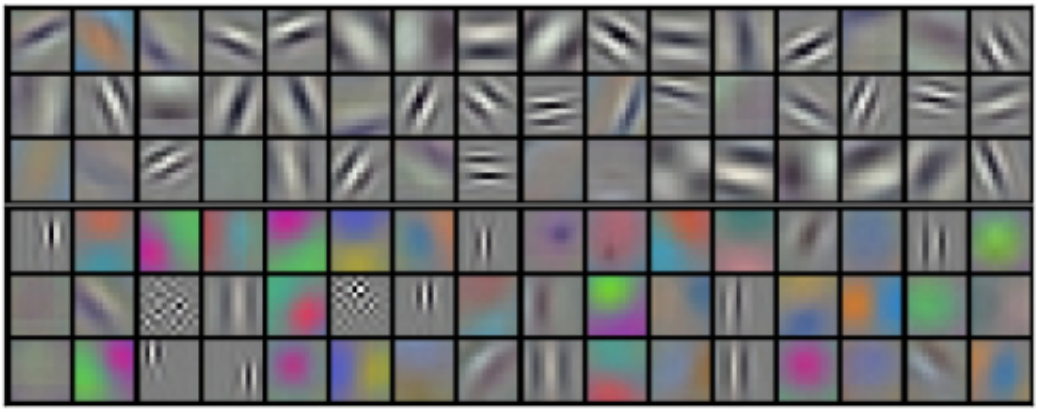

FIGURE 4 – Représentation dans l'espace RGB des paramètres de la première couches de convolution de AlexNet.

images d'entrée en leur soustrayant l'« image moyenne », c'est à dire la moyenne des images du jeu de données, et « réduire »en les divisant par l'image d'écart type moyen.

Questions :

— Combien de couches de convolution (Conv2d), fully-connected (Linear) ?
— Combien de couches de neurones totales ?
— Qu'est ce qu'un module Linear, Conv2d, ReLU, MaxPool2d ? Combien de paramètres ?
— Pourquoi peut-on visualiser les paramètres de la première convolution dans l'espace RGB et pas les autres ? Pourquoi ce n'est tout de même pas facile à interpréter sur VGG16 ?
— Intuitivement, quelles concepts sémantiques sont présents dans les premières couches

de convolution ?

— Intuitivement, quelles concepts sémantiques sont présents dans la dernière couche de classification (Linear) ?

— Intuitivement, quelles concepts sémantiques sont présents dans l'avant dernière couche de classification (Linear) ?

6 Extraction de features

Dans Chatfield *et al.* (2014), on utilise le réseau de neurones de façon à produire des représentations riches (*features*) des images. Plus précisément, l'output de la couche 19 (celle précédant les sorties finales du réseau).

Note : dans l'article, les auteurs appliquent une normalisation L2 sur les features en entrée du classifieur. C'est-à-dire que pour une image dont le vecteur de features est $x = (x_1, x_2, ..., x_p)$, on applique la transformation :

$$x_i^{(norm)} = \frac{x_i}{||x||_2} = \frac{x_i}{\sqrt{\sum_{i=1}^{p} x_i^2}}$$

Vous devez choisir un réseau (AlexNet pour commencer) et une couche de neurones d'extraction.

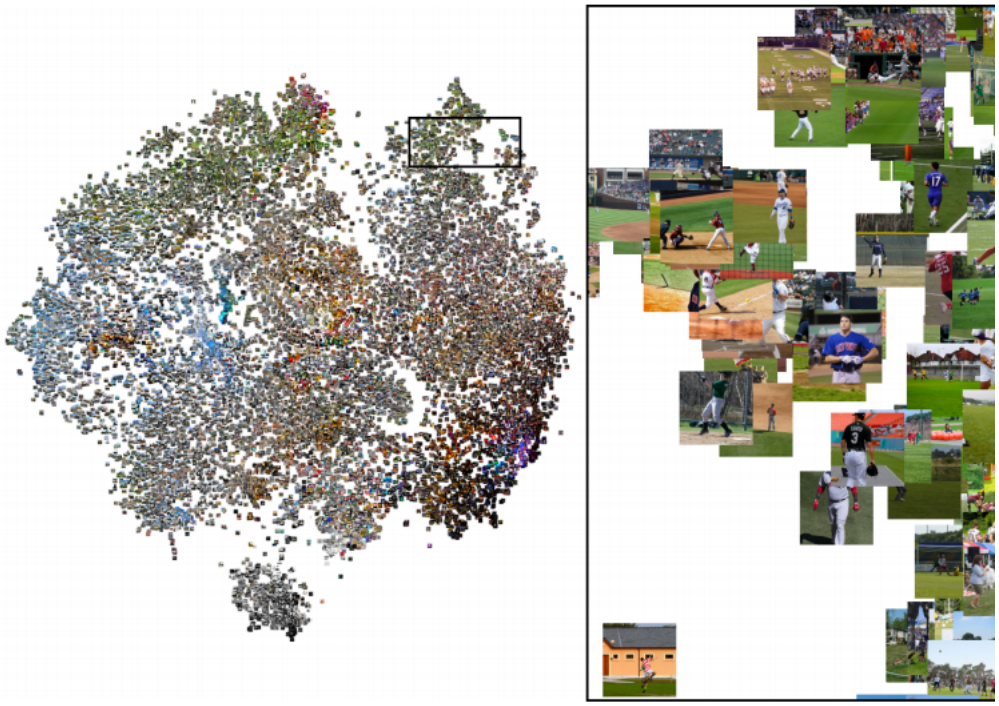

FIGURE 5 – Représentation dans un espace 2D avec l'algorithme t-SNE des images d'ImageNet à partir des features extraites de l'avant dernière couches de VGG16.

Questions :

— Quelles couches choisir ? Avant ou après la ReLU ?
— Quelle est l'intérêt de la normalisation L2 ?

7 Entrainement d'un modèle de classification

Les features extraites grâce au réseau de neurones sont des représentations des images. En connaissant les labels des images d'apprentissage, on peut apprendre, dans l'espace des features, une frontière de décision entre d'une part les features des images de label positif et d'autre part les features des images de label négatif. C'est le rôle du classifieur Machine à Vecteur de Support (Support Vector Machine). On utilise ici un noyau linéaire.

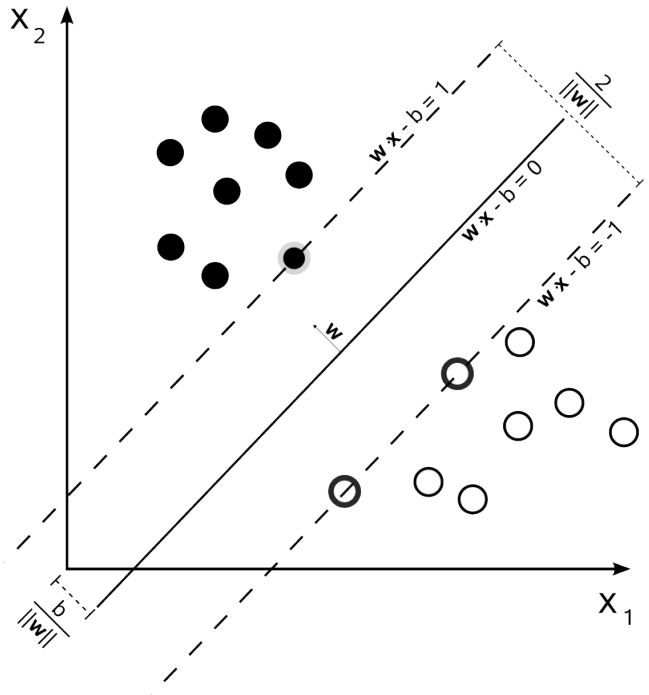

FIGURE 6 – Hyperplan de marge maximum et marges pour un Support Vector Machine. Les exemples blancs appartiennent à la classe 1. Les exemples noirs appartiennent à la classe 0. Les exemples sur la marge sont appelés vecteurs de support.

Le mAP *(Mean Average Precision)* est une mesure classique d'évaluation qui permet de prendre en compte la confiance avec laquelle la classe prédite a été determinée. C'est la moyenne des scores d'*Average Precision* (AP) calculés pour chaque classe.

L'apprentissage se fait sur les features extraites des images de *train* et de *val*. La fonction `train` permet d'apprendre la frontière de décision. On peut ajouter des options, comme par exemple changer la valeur de la constante de régularisation C.

Durant la première étape d'apprentissage, plusieurs classifeurs multi-labels sont appris

sur l'ensemble de train et évalués sur l'ensemble de val afin de trouver les hyperparamètres optimaux tels que le C, les features extraites, la présence ou l'absence de normalization $L2$ des features, etc.

Durant la seconde étape d'apprentissage, un classifieur multi-labels est entrainé (*une seule fois*) avec le jeu d'hyperparamètres optimaux trouvés durant la première étape. Ce classifieur est entrainé sur train+val et validé sur test.

Note : Des méthodes plus ou moins couteuses que celle présentée ici peuvent être employées. Toutes doivent garantir que l'ensemble d'évaluation (test) n'ait jamais été vu durant le processus d'apprentissage. Afin de comparer deux modèles, il est obligatoire de les entrainer avec la même méthode.

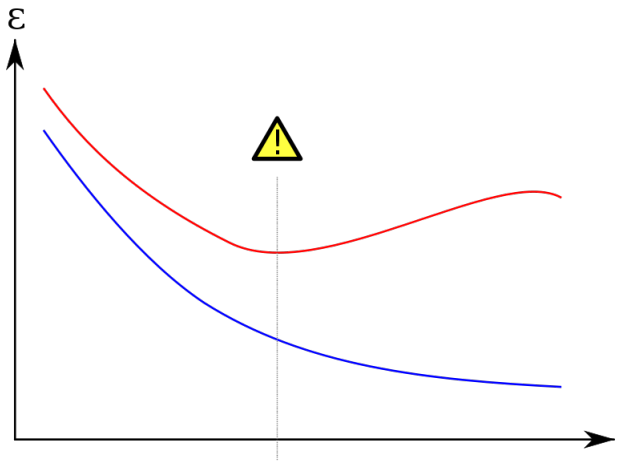

FIGURE 7 – Surapprentissage (overfitting) dans un apprentissage supervisé. En ordonné l'erreur, en abscisse l'hyperparamètre de régularisation. En rouge, l'erreur sur l'ensemble de validation. En bleu, l'erreur d'apprentissage. Si l'erreur de validation (val ou test) augmente alors que l'erreur d'apprentissage (train ou trainval) continue à diminuer, il y a un risque de surapprentissage.

Questions :

— Qu'est ce qu'un Support Vector Machine ?
— Quel est l'effet de la constante de régularisation du SVM ?
— Qu'est ce que l'Average Precision, l'Accuracy ?
— Pourquoi utilise-t-on la métrique d'Average Precision et pas d'Accuracy ?
— Quelle est la différence entre un classifieur multiclasses et multilabels ?
— Quelle est la différence entre un classifieur multilabels "one versus rest" et "one versus one" ?
— Quelle est la différence entre LinearSVC et SVC ?
— Quelle est l'utilité d'un noyau non linéaire pour le SVM ? Est-ce utile dans notre contexte ?

7.1 Approfondissement

Réglage de l'apprentissage du SVM : Effectuez des tests avec différentes valeurs de C pour l'apprentissage du SVM.

Etude du temps d'exécution : Relevez le temps nécessaire à l'accomplissement des différentes étapes du processus implémenté.

Autres features deep : Effectuer des tests avec des features extraites à d'autres niveaux du réseau de neurones.

Autres architectures de réseaux : Reproduire les résultats obtenus en appliquant d'autres réseaux pré-entraînés et comparer avec les résultats obtenus.

8 Références et liens

Bibliothèques

- PyTorch : `http://pytorch.org`
- TorchVision : `https://github.com/pytorch/vision`
- ScikitLearn : `http://scikit-learn.org`

Bases de données

- ImageNet : `http://www.image-net.org`
- VOC 2007 : `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html`

Liens utiles

- Explication de l'Average Precision : `https://sanchom.wordpress.com/tag/average-precision`
- Article disponible sur `http://www.robots.ox.ac.uk/~vgg/publications/2014/Chatfield14`
- Pour les curieux, quelques informations sur t-SNE : `https://distill.pub/2016/misread-tsne`
- Le SVM expliqué pour des enfants de 5 ans (anglophones) : `https://www.reddit.com/r/MachineLearning/comments/15zrpp/please_explain_support_vector_machines_svm_like_i`

Références

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman "Return of the Devil in the Details : Delving Deep into Convolutional Networks, " in *BMVC*, 2014.