TP réseaux de neurones

Matthieu CORD, Micael CARVALHO, Thomas ROBERT, Rémi CADENE
Multimedia

2016

1 Objectif du TP

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.

On s'intéresse plus particulièrement dans ce papier à des réseaux de neurones pré-entraînés sur des données externes (ici, ImageNet). La structure du réseau est alors définie par avance, et les poids des différents filtres de convolution sont appris sur des images du dataset externe.

En appliquant le réseau à chaque image du dataset cible, on obtient les outputs de chaque couche : les features décrivant l'image sont alors l'output de la couche voulue. En pratique, on s'intéressera à l'output de la dernière couche avant la couche calculant les scores (i.e. l'output de la couche 19).

On s'intéressera donc particulièrement à la ligne (o) : features deep extraits grâce à un réseau pré-entraîné (VGG-M avec MatConvNet) + classification grâce à un SVM linéaire (liblinear).

2 Composants de la chaîne de traitement

Commencons par des rappels et notes sur les différents composants de la chaine de traintement utilisée par Chatfield *et al.* (2014) que vous devrez reproduire.

2.1 Jeu de données d'entrée : VOC 2007

PASCAL VOC 2007 est un dataset constitué de 9963 images multi-label, stockées dans le dossier `/opt/matconv/VOCdevkit/VOC2007/JPEGImages/`. Il faut donc considérer le pro-

blème comme une succession de problèmes binaires *one vs 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 ;
— les images de *test* ne seront utilisées que pour évaluer le SVM ainsi appris.

Pour chaque classe, un fichier texte est disponible sous la forme `class_set.txt` , où *set* est la chaîne de caractères *train*, *val*, *trainval* ou *test*. Les labels y sont stockés par ligne sous la forme : `id_image label` , où *label* est dans $\{+1, -1\}$. La fonction `[ids, labels] = textread(labelfile,'%s %d')` permet de lire ce fichier :

— `labelfile` est le chemin du fichier de labels ;
— `ids` est la liste des noms des fichiers image (sans l'extension) ;
— `labels` est le vecteur des labels correspondants.

Ces fichiers sont dans le dossier `/opt/matconv/VOCdevkit/VOC2007/ImageSets/Main/`.

NB : 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.*

2.2 Réseau de neurones & extraction de features : *MatConvNet*

MatConvNet fournit de nombreux outils pour manipuler les réseaux de neurones sous MATLAB.

2.2.1 Représentation du réseau de neurones

Un réseau de neurones y est représenté comme une structure à plusieurs couches. Chaque couche de la structure représente alors une couche du réseau de neurones. Plusieurs types de couches sont implémentés, dont notamment celles dont nous nous servirons pour ce travail :

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

Les poids pré-entraînés des réseaux décrits dans Chatfield *et al.* (2014) sont disponibles dans le dossier : `/home/sasl/shared/ei-se/tp_multimedia/pretrained_models/`.

2.2.2 Entrée du réseau de neurones

Il est courant d'appliquer un prétraitement aux entrées d'un réseau de neurones (dans notre cas des images), afin de les « normaliser ». 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 images d'entrée en leur soustrayant l'« image moyenne », c'est à dire la moyenne des images du jeu de données.

2.2.3 Extraction de *features*

Pour chaque image, on peut calculer l'output de chaque couche d'un réseau : c'est le rôle de la fonction *vl_simplenn*. Elle est disponible dans le dossier `/opt/matconv/MatConvNet/matlab/`. Le résultat obtenu est alors une structure contenant dans chaque couche $i+1$ l'output de la couche i du réseau de neurones (cf figure 1).

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

Comme dans l'article, pensez à appliquer 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 :

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

2.2.4 Installation de MatConvNet

MatConvNet est disponible dans le dossier `/opt/matconv/MatConvNet/`, néanmoins, il est necessaire de compiler cette librairie pour pouvoir l'utiliser pendant ce TP. Pour ce faire, il suffit de naviguer au chemin ci-dessus (commande `cd` dans l'invite de commande MATLAB) et de lancer le script de compilation en tapant dans la console MATLAB la commande : `run matlab/vl_compilenn`.

2.2.5 Exemple de script

Regardez le script fourni nommé `test.m`. Il permet de charger les modules de MatConvNet et de réaliser de nombreuses opérations décrites précédemment : charger un réseau (VGG-M ici comme dans l'article), charger et normaliser une image, calculer les sorties des couches du réseau et les afficher.

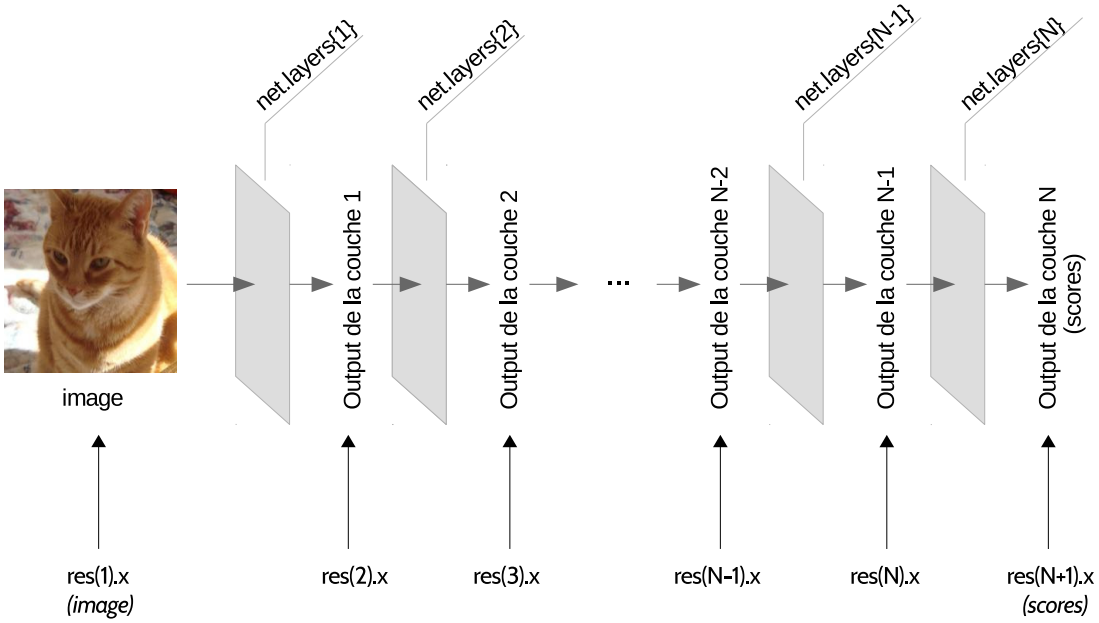

FIGURE 1 – Représentation du réseau (net) et de l'output de la fonction *vl_simplenn* (res)

Lancez, analysez, testez ce script pour vous familiariser avec MatConvNet. Essayez de l'appliquer à différentes images de VOC 2007. Vous pouvez aussi regarder les sorties à diverses couches.

2.3 Classifieur SVM : *liblinear*

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 SVM. On utilise ici un classifieur linéaire.

2.3.1 Installation de liblinear

On utilisera la librairie *liblinear* pour cette étape. La librairie est disponible dans le dossier suivant : `/opt/matconv/liblinear-2.01/matlab/`.

Pour l'utiliser dans MATLAB, il suffit d'ajouter le dossier ci-dessus au *path* MATLAB, par exemple en appelant au début de votre code la fonction `addpath(...)`.

2.3.2 Apprentissage sur *train* et *val*

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. Elle s'utilise comme suit : `model = train(train_labels, train_features)` où

- `train_labels` est un vecteur de taille $nb\_images\_train \times 1$ de type `double`
- `train_features` est une matrice de taille $nb\_images\_train \times dimension\_features$ de type `sparse(double)`.

On peut ajouter des options, comme par exemple changer la valeur de la constante de régularisation C en appelant la fonction : `model = train(train_labels, train_features, sprintf(' -c %f',C))`. Pour plus de détails, taper `train` dans l'invite de commande MATLAB.

2.3.3 Prediction sur *test*

On teste alors le modèle sur les features extraites des images de l'ensemble de test : c'est le rôle de la fonction `predict`. Elle s'utilise comme suit : `[predicted_labels, accuracy, scores] = predict(test_labels, test_features, model)` où

- `test_labels` est un vecteur de taille $nb\_images\_test \times 1$ et de type `double` ;
- `test_features` est une matrice de taille $nb\_images\_test \times dimension\_features$ et de type `sparse(double)` ;
- `model` est le modèle appris grâce à la fonction `train` ;
- `predicted_labels` est le vecteur des labels predits pour les images d'entrée ;
- `accuracy` est le taux de bonne classification, c'est à dire le rapport du nombre d'images bien classées sur le nombre total d'images en test ;
- `scores` est le vecteur des scores du classifieur pour les images d'entrée.

Pour plus de détails, taper `predict` dans l'invite de commande MATLAB.

2.4 Évaluation des performances : métrique mAP

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.

Une implémentation du calcul de l'average precision pour une classe et un ensemble d'images données est proposée dans le dossier `/home/sasl/shared/ei-se/tp_multimedia`. Elle s'utilise de la manière suivante :
`ap = compute_class_AP(labels_GT, scores)` où

- `labels_GT` est le vecteur des vrais labels des images ;

— `scores` est le vecteur des scores de classification obtenus ;
— `ap` est l'average precision de l'ensemble testé (pour une classe donnée).

Pour pouvoir utiliser cette fonction depuis votre code, il suffit d'ajouter le chemin ci-dessus dans le *path* MATLAB comme pour *liblinear*.

3 Travail à réaliser

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

On s'intéressera particulièrement à la ligne (o) : features deep extraits grâce à un réseau pré-entraîné + classification grâce à un SVM linéaire.

— Chargement et pré-traitement des images
— Extraction des features deep grâce aux poids pré-appris
— Apprentissage d'un SVM linéaire
— Test / calcul du mAP

3.1 Conseil pour l'implémentation

Voir détail en partie 3.3.

1. Implémenter la chaîne de classification pour une classe (par exemple la classe `aeroplane`).
   - Normalisation des images (taille fixe + soustraction de l'image moyenne)
   - Extraction des features deep sur toutes les images avec le réseau `imagenet-vgg-m` (output de la couche 19)
   - Apprentissage du SVM sur les features deep normalisées L2 extraites des images de *train + val* (hormis les images difficiles)
   - Test du modèle sur les features deep normalisées L2 extraites des images de *test* (hormis les images difficiles)

   Vérifiez que pour la classe `aeroplane`, vous obtenez un average precision de 89.5%.
2. Etendez ensuite à toutes les classes. Le fichier `classes_PASCALVOC2007.mat` (`dossier/home/sasl/shared/ei-se/tp_multimedia`) contient la liste des classes de VOC 2007.

3.2 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 (dossier `/home/sasl/shared/ei-se/tp_multimedia/pretrained_models`).

3.3 Emploi du temps et travail à rendre

Ce TP se décompose en 4 séances de 4h (13h30 - 18h) détaillées ci-dessous.

A la fin de chaque séance (sauf la dernière), vous devrez rendre un compte rendu succinct au format papier décrivant le travail effectué pendant la séance. Ces compte-rendu représenteront 25% de note de TP. Un compte rendu final et détaillé sera à rendre après la quatrième séance et représentera 75% de la note.

- **mardi 4 octobre** : prise en main MATLAB / MatConvNet
  - Compréhension du détail du réseau / des outputs
  - Chargement différents réseaux / différentes images
  - Calcul de la classe prédite / du score sur différentes images

- **mardi 18 octobre** :
  - Calcul de la classe prédite / du score sur différentes images de VOC 2007
  - Extraction des features sur toutes les images de VOC 2007
  - Stockage de la matrice des features dans un fichier grâce à la fonction MATLAB `save`

- **mardi 8 novembre** :
  - Implémenter l'apprentissage et le test d'un SVM sur VOC 2007 pour la classe `aeroplane` ; vérifier que, pour cette classe, vous obtenez un AP d'environ 89.5%.
  - Etendre le processus à toutes les classes

- **mardi 22 novembre** : compte-rendu du travail effectué sur l'ensemble du TP, à rendre à la fin de la séance
  - Calculer le mAP pour toutes les classes, pour un paramétrage donné.
  - Implémenter les approfondissements proposés.

- **mardi 13 decembre** (à confirmer) : exposé oral de 5 min par binôme où vous exposerez votre avancement et vos commentaires sur chacune des étapes de votre travail.

4 Références et liens

Liens pour téléchargements

— Logiciels :

- MatConvNet :
  - — package : `https://github.com/vlfeat/matconvnet`
  - — instructions d'installation : `http://www.vlfeat.org/matconvnet/install/`
  - — poids pré-entraînés : `http://www.vlfeat.org/matconvnet/pretrained/`
- LibLinear : `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

— Base de données :

- VOC 2007 : `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html`

— Article disponible sur `http://www.robots.ox.ac.uk/~vgg/publications/2014/Chatfield14/`.

Référence

[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.