RDFIA TP2 : Prise en main de MatConvNet

Taylor Mordan, Thomas Robert, Matthieu Cord

2 novembre 2016

Pour les TP sur les réseaux de neurones, vous devrez rendre :
- Un compte rendu à la fin de chaque séance (manuscrit ou par mail à `taylor.mordan@lip6.fr` ou `thomas.robert@lip6.fr` selon votre encadrant de TP, les compte-rendus en retard ne seront pas acceptés). Ce compte-rendu devra décrire le travail effectué pendant la séance et les résultats importants obtenus.
- Un compte rendu global une semaine après la dernière séance (par mail), dont le but est de reprendre au propre le travail effectué pendant l'intégralité des séances de TP, en prenant du recul sur ce qui a été fait. Ce compte rendu représentera la majorité de la note de TP.

Les données et une version numérique du sujet sont accessibles
à l'adresse `http://webia.lip6.fr/~robert`

1 Objectifs

L'objectif de ce TP est de se familiariser avec les structures de données et les fonctions principales de MatConvNet, une bibliothèque de réseaux de neurones convolutionnels sous MATLAB. Nous verrons également comment utiliser un modèle existant pour obtenir une prédiction de classe sur une image test. Toutes les notions vues dans ce TP serviront de base pour les TPs suivants !

2 Représentation et fonctionnement d'un réseau de neurones

MatConvNet fournit de nombreux outils pour manipuler les réseaux de neurones convolutionnels sous MATLAB. Un réseau de neurones y est représenté comme une structure où chaque élément représente une couche du réseau (voir Figure 1). Plusieurs types de couches sont implémentés et peuvent être utilisés directement, dont notamment celles dont nous nous servirons pour ce TP :
- couche de convolution ;
- couche de *pooling* ;
- couche ReLU ;
- couche de normalisation ;
- couche SoftMax.

Pour chaque image, il faut calculer successivement la sortie de chaque couche d'un réseau à partir des précédentes : c'est le rôle de la fonction `vl_simplenn` (voir Figure 1). Nous allons voir les étapes pour charger un modèle, préparer les images, puis calculer la classe prédite.

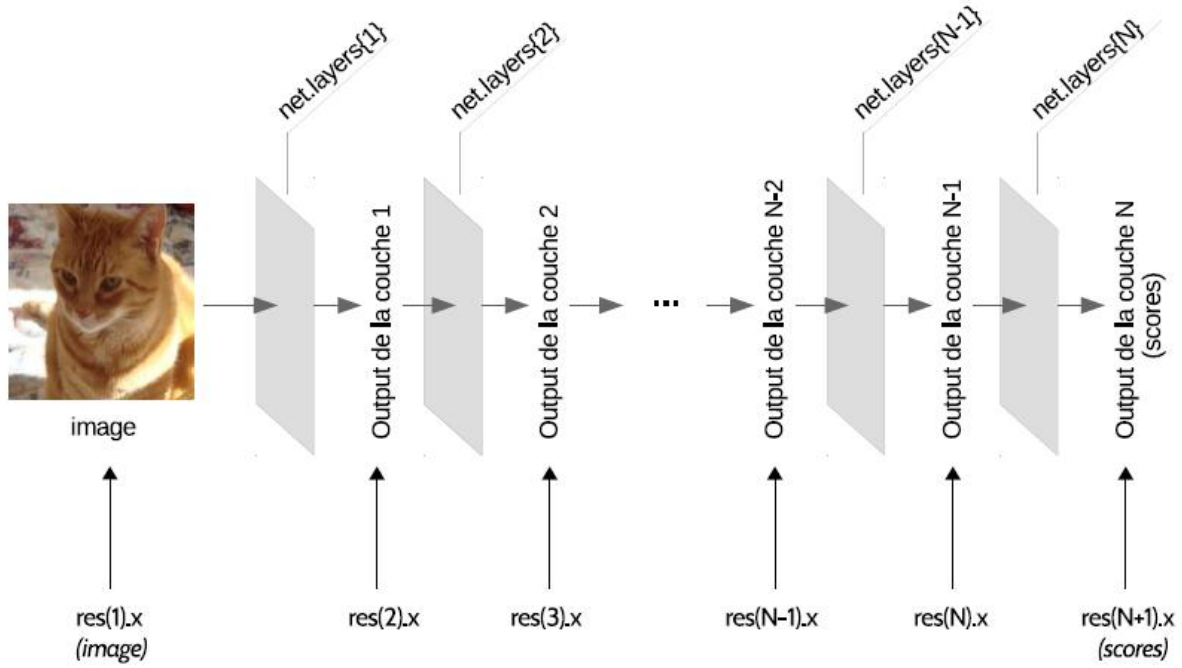

FIGURE 1 – Représentation du réseau (`net`) et de la sortie de la fonction `vl_simplenn` (`res`).

1. Créer un dossier de travail que nous appellerons `RDFIA` et s'y déplacer. Créer un script `rdfia_matconvnet.m` dans ce répertoire. Tout sera fait à la suite dans ce script. Il commencera par les instructions pour initialiser MatConvNet (ce qui sera à faire à chaque fois que MATLAB est lancé).

```
clc;
clear all;
close all;
% Initialisation de MatConvNet sous MATLAB (à faire à chaque
    nouvelle session)
run /usr/local/matconvnet-1.0-beta23/matlab/vl_setupnn;
```

2. Charger le réseau CNN-F (voir l'article [1]) ; nous l'appellerons `net`. La structure du réseau est décrite dans l'article [1]. Il est possible de visualiser les caractéristiques des couches du réseau avec la fonction `vl_simplenn_display`.

```
% Chargement du modèle et conversion vers la version actuelle
    de MatConvNet
net = load('/usr/local/imagenet/imagenet-vgg-f.mat');
net = vl_simplenn_tidy(net);
% Visualisation des informations sur les couches du réseau
vl_simplenn_display(net, 'inputSize', [224 224 3 1])
```

3. Charger une image (commencer par `peppers.png`, `pears.png` et `onion.png` disponibles par défaut) et la préparer pour l'entrée du réseau `net`.

```
% Charger et préparer l'image
```

```
im = imread('peppers.png');
im_ = single(im); % Note : valeurs entre 0 et 255
im_ = imresize(im_, net.meta.normalization.imageSize(1:2));
im_ = im_ - net.meta.normalization.averageImage;
```

4. Calculer la sortie `res` des différentes couches du réseau `net` (CNN-F dans l'article [1]). Pour cela, il faut appliquer la fonction `vl_simplenn`. Le résultat `res` est alors une structure où chaque élément i+1 contient la sortie de la couche i du réseau de neurones (cf. Figure 1).

```
% Calculer la prédiction du modèle
res = vl_simplenn(net, im_);
```

5. Afficher le score et la classe prédite (parmi les 1000 disponibles) :

```
% Afficher la classe prédite
scores = squeeze(gather(res(end).x));
[bestScore, best] = max(scores);
figure(1); clf; imagesc(im);
title(sprintf('%s (%d), score %.3f',...
   net.meta.classes.description{best}, best, bestScore));
```

3 Compréhension détaillée du réseau et des sorties

1. Lister le contenu de tous les traitements de `net` et commenter le reste des informations présentes. Quelle est l'utilité de chaque type de couche ?
2. Afficher l'image en entrée du réseau. Que s'est-il passé entre cette image et l'originale ? Expliquer les étapes du pré-traitement. Pourquoi n'est-il pas possible de mettre une image $300 \times 300 \times 3$ en entrée ?
3. Lister les tailles de toutes les structures de `res`.
4. Expliquer les tailles de `res(1)` et `res(2)` pour CNN-F en vous aidant de l'article [1] (Table 1) et sachant que *stride* est le pas de convolution. Trouver une relation générale liant les tailles des entrées et des sorties des couches de convolution en fonction des paramètres de la couche.
5. Afficher des images correspondant à différentes cartes obtenues après la première convolution. Comment interpréter ces cartes ?
6. Afficher les scores des dernières couches et commenter l'effet de la ReLU et du SoftMax.
7. Visualiser les résultats obtenus pour différentes images de votre choix.
8. Bonus : à l'aide de la documentation en ligne de MatConvNet, expliquer comment et avec quelle(s) fonction(s) est calculée la passe *backward*. Comparer cette méthode avec ce qui a été fait au TP1.

Références

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details : Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.