Deep Learning Practical Work 3-a
Transfer Learning through feature extraction from a CNN

Nicolas Thome Guillaume Couairon Mustafa Shukor Asya Grechka
Matthieu Cord

Homework

- For all practical works, contact the supervisor of your group: `nicolas.thome@sorbonne-universite.fr`, `mustafa.shukor@isir.upmc.fr`, `guillaume. couairon@ gmail.com` and `aagrechka @gmail.com`
- Email object must be `[RDFIA][Homework-3]`.
- After doing 3-d, you will need to finish by yourself at home the whole homework 3 (3-a, 3-b, 3-c, and 3-d). You will send by email this homework. You are encouraged to do it by group of two (please write both names in the email!)
  - You need to provide answers to all questions in one single PDF file. You can include inside the PDF code snippets (not a whole file!), graphs, or images that you deemed necessary.
  - Don't write long paragraphs of bullshit, we'll just get tired of it and be more severe.
  - Questions with a "Bonus" mark are optional. Questions with a ★ are worth double points.

Data, code, and PDF version of this file are available at
`https://rdfia.github.io`

Goals

The goal of this TP is to become acquainted with a well-known and commonly-used deep CNN network: VGG16 (Simonyan & Zisserman, 2015). Then, the method described by Chatfield et al. (2014) will be used with this network on the *15 Scene* dataset. We will particularly look at the strategy to extract *deep* features thanks to a pre-trained network, and we will use these features in a classic classification scheme using linear SVM.

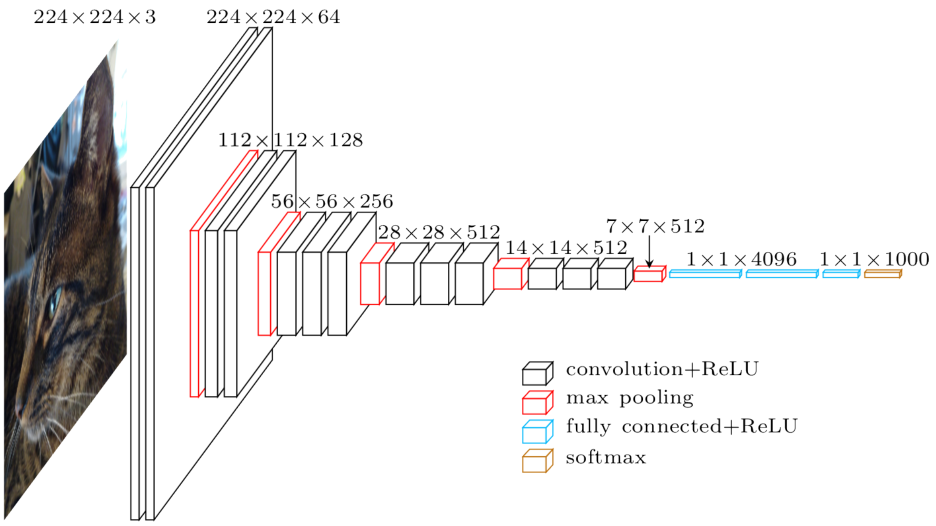

Figure 1: VGG16 Network

Section 1 – VGG16 Architecture

The convolutional network VGG16 (Simonyan & Zisserman, 2015, voir Figure 1) is a classification model trained on the ImageNet (Russakovsky et al., 2015) dataset. This dataset contains over 1 million images categorized in 1000 classes. Its architecture consists in five blocks, each composed of:

- two or three convolutional layers preserving the spatial dimensions;
- a pooling layer which halves the spatial dimensions.

Then, the number of feature maps are doubled at each following block. The end of the network contains three fully-connected layers.

The pre-trained VGG16 network can be loaded in PyTorch with the following command:

```python
import torchvision
vgg16 = torchvision.models.vgg16(pretrained=True)
```

The images given as input in the network need to be resized to 224×224 pixels, and normalized with the mean $\mu = [0.485, 0.456, 0.406]$ and the standard deviation $\sigma = [0.229, 0.224, 0.225]$ (calculated on the training set of ImageNet). The list with the 1000 classes from ImageNet can be obtained with a provided dictionary. Loading the dictionary and an image can be achieved with the following code:

```python
import pickle
from PIL import Image
import numpy as np

imagenet_classes = pickle.load(open('imagenet_classes.pkl', 'rb')) # load classes

img = Image.open("cat.jpg")
img = img.resize((224, 224), Image.BILINEAR)
img = np.array(img, dtype=np.float32) / 255
img = img.transpose((2, 0, 1))
# TODO preprocess image
img = np.expand_dims(img, 0) # transform into a batch containing one image
```

```
x = torch.Tensor(img)
y = ... # TODO forward pass
y = y.numpy() # transform to numpy array
# TODO obtain predicted class and its score
```

Questions

1. ★ Knowing that the fully-connected layers account for the majority of the parameters in a model, give an estimate on the number of parameters of VGG16 (using the sizes given in Figure 1).
2. ★ What is the output size of the last layer of VGG16? What does it correspond to?
3. **Bonus :** Apply the network on several images of your choice and comment on the results.
4. **Bonus :** Visualize several activation maps obtained after the first convolutional layer. How can we interpret them?

Section 2 – Transfer Learning with VGG16 on 15 Scene

2.1 Approach

General Principle The principle of the approach is rather simple and consists in two steps: we will first use a pre-trained neural network to perform *feature extraction* which we then use to train a classification network.

The first step can be seen as an alternative on the SIFT + Bag of Words (BoW) approach. The goal is the obtain a vector representation for every image which we can then later use for a variety of tasks, particularly image classification.

The principle of *feature extraction* consists in producing this image representation using a network which was trained to solve a different task than ours. In our case, the VGG16 network was trained for the task of image classification on the ImageNet dataset. To produce the image representations, we will use the output of intermediate layers of the chosen pre-trained network.

Principle for our TP In the case of our TP, we will use the VGG16 network, and we will represent each image as the vector outputted from the `relu7` layer (the ReLU just before the classification layer). We will then train an SVM classifier on each class of the *15 Scene* dataset to solve our classification task.

Questions

5. ★ Why not directly train VGG16 on 15 Scene?
6. ★ How can pre-training on ImageNet help classification for 15 Scene?
7. What limits can you see with feature extraction?

2.2 Feature Extraction with VGG16

To extract the features at the *relu7* layer of VGG16, create a new class `VGG16relu7` where you will copy the VGG16 layers up until the *relu7* (see the code below). The output of this new network will be the desired features. To extract the features from another layer, you can modify the `classifier` field (or potentially `features`) to delete more layers.

The input images to the network need to be resized to the correct shape (with the correct number of channels!) and normalized with the mean and standard deviation given above.

Using this procedure, create the matrices $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$ in which each row corresponds to a feature vector. Each feature vector will likewise be normalized with the L2 norm.

Questions

8. What is the impact of the layer at which the features are extracted?
9. The images from 15 Scene are black and white, but VGG16 requires RGB images. How can we get around this problem?

```
class VGG16relu7(nn.Module):
    def __init__(self):
        super(VGG16relu7, self).__init__()
        # Copy the entire convolutional part
        self.features = nn.Sequential( *list(vgg16.features.children()))
        # Keep a piece of the classifier: -2 to stop at relu7
        self.classifier = nn.Sequential(*list(vgg16.classifier.children())[:-2])

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```

2.3 Training SVM classifiers

Using the *features* ($\mathbf{X}_{train}, \mathbf{X}_{test}$) and the corresponding *targets* (y_{train}, y_{test}), we will train a multi-class classifier (composed of many binary linear SVM classifiers in *one-versus-all*) with these elements. This work has already been done previously; you will use `sklearn.svm.LinearSVC` from Scikit Learn.

```
from sklearn.svm import LinearSVC
svm = LinearSVC(C=1.0)
```

Train a multi-class SVM classifier on the training set with $C = 1$ and evaluate its classification score (*accuracy*) on the test set using the *deep features* previously extracted. For this, the functions `fit` and `score` from `LinearSVC` will be used to train the SVM and calculate its performance.

Questions

10. Rather than training an independent classifier, is it possible to just use the neural network? Explain.

2.4 Going further

Once this classification sceme works, you can study the effect of two (or more) choices on the performances or training time. Some possible examples are listed below but you are free to find others!

- Change the layer at which the *features* are extracted. What is the importance of the depth of this layer? What is the representation size and what does this change?
- Try other available pre-trained networks. What are the differences between these networks?
- Tune the parameter C to improve performance.
- Instead of training an SVM, replace the last layer of VGG16 with a new fully-connected layer and continue to train the network on 15 Scene (with or without propogating the gradients to the rest of the network).
- Look into methods for dimensionality reduction before classification and their impact on performance and execution time.

Question

11. For every improvement that you test, explain your reasoning and comment on the obtained results.

References

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.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Internationale Journal of Computer Vision (IJCV)*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.