Practical Work 06 – 28/03/2024
Conv Neural Networks - part 2

Objectives

The objective of this PW is to understand some more advanced methods to train and use Convolutional Neural Networks (CNN) including data augmentation and techniques of visualisation inside the network. Another objective is to review classical deep architectures and understand the recent strategies to compose such architectures. **You may realise this PW with Keras/TF or Pytorch.**

Submission

— **Deadline** : Wednesday 10th of April 2024, 12am (noon)

— **Format** : Zip with report and/or iPython notebook.

Exercise 1 Data Augmentation

Use the notebook `CIFAR10-CNN_data_augmentation_stud_TF.ipynb` available on Moodle as starting point.

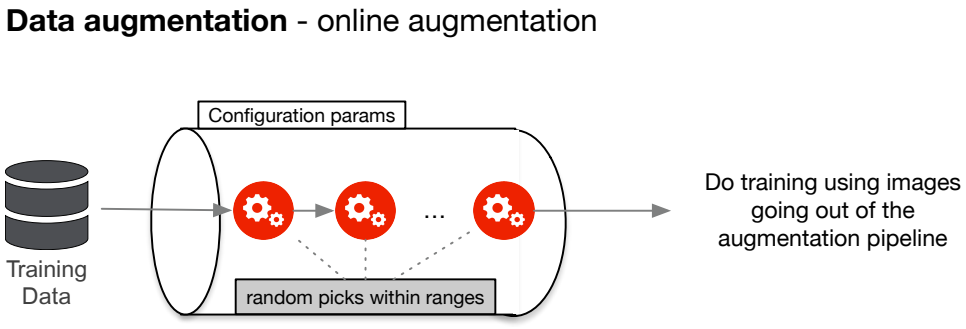

FIGURE 1 – Online data augmentation pipeline

a) Train a CNN

Define a CNN with the following structure : `CONV(32F,same)-RELU-CONV(32F,same)-RELU-MAXP(2)-CONV(32F,same)-RELU-MAXP(2)-DENSE`. Train the network using 10 epochs and batches of 128 images. Use a `categorical_crossentropy` loss and the `adam` optimizer.

b) Train the CNN with data augmentation

Re-read the section of the slides explaining the principles of data augmentation for images. Either Keras or Pytorch will allow you to use an *online* data augmentation strategy as illustrated on Figure 1. Using the example given for the FashionMNIST dataset (cf. slides), implement a similar data augmentation strategy for CIFAR10.

You may try with different strategies and hyperparameter values of the data augmentation tools provided in Keras or Pytorch.

a) Report the accuracy on the train set and on the test set for your different experiments. Do you observe an improvement using data augmentation ?
b) Compare the evolution of the loss through the training epochs, with and without using data augmentation. Comment your observations.
c) If you tried with different data augmentation strategies, which one seems to give the best results ?

Exercise 2 Visualisation of activations

The objective is here to visualise the different activation maps in the network previously trained. The Figure 2 illustrates the principle for the first CONV layer on the first 6 filters of a given network.

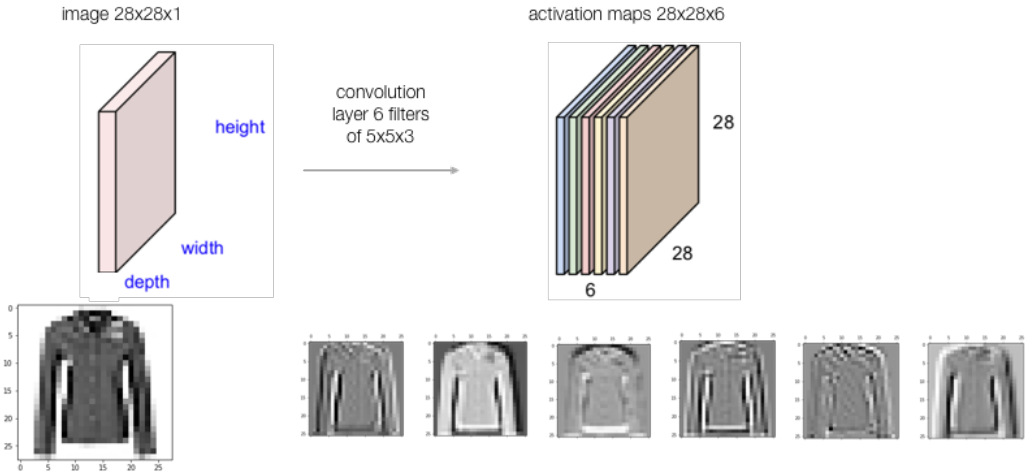

FIGURE 2 – Visualisation of activations of a CONV layer

Using the best of your network previously trained on CIFAR10 in exercise 1, implement a visualizer for the activations at different layer outputs.

a) Read again the example of the visualisation presented in the class.
b) Implement a code to visualise all the filters at a given layer. Hints : use subplots to have a grid of images, use `for` loops to avoid code repetition.
c) For a given input image (e.g. `X_train[12]`), visualise the different activations maps of your network : outputs of CONV, RELU, MAXP. Comment on what you see.

Exercise 3 Optional : Deeper Models

Let's play here with a deeper CNN model on CIFAR10 using a structure inspired by VGGNet : `[[Conv2D` → `relu` → `BN]*3` → `MaxPool2D` → `Dropout]*4` → `Flatten` → `Dense` → `Dropout` → `Out`. In this structure, `BN` means Batch Normalisation. You can try different options but a configuration with blocks of 64 filters of size 3 with same padding and stride 1, max pooling of size 2 and stride 2, dropouts of 0.2 and a dense layer of 256 neurons should bring your performance around 80-85% on CIFAR10.

You probably need to use a GPU to train such networks and play with different settings. You may use freely available GPUs on Colab, see https://colab.research.google.com/ [1]. Report your best performances with and without data augmentation as in Exercise 1. What are your observations.

Exercise 4 Analysis of a Deep Architecture

The analysis of the deep CNN architectures we did in the class covered the evolution observed on the ImageNet LSVRC competition until 2017. New architectures were introduced after that, such as for exemple the **Inception-v4**, **XCeption**, **EfficientNet**, etc. Other strategies based on deep CNNs have also emerged for other tasks such as *image detection* (finding bounding boxes around the objects of interest), such as **Yolo** (v1 to v6).

a) Pick one of these new architectures that you find interesting (Inception-v4, XCeption, EfficientNet, Yolo, or another one).
b) List and provide the reference(s) explaining the architecture (at least the reference to the original paper(s) presenting the architecture).
c) Read the article(s) up to the point you have a general understanding of their strategy [2].
d) Re-explain in few phrases what you understood from the architecture doing comparison with the architectures presented in the class.

Exercise 5 Optional : Review Questions

a) Explain the notion of hierarchical features with CNNs.
b) Explain 2 strategies to visualise the modelling taking place in CNNs.
c) What do we try to fight when using data augmentation ?
d) What are the different implementation strategies for data augmentation ?
e) Can we say that data augmentation is a form of regularization allowing to limit overfitting ?
f) Explain the main differences for the deep architectures seen in class : AlexNet, VGGNet, GoogLeNet, ResNet. What were their intuitions when putting together such architectures ?

1. To activate the gpu on Colab, go to Edit > Notebook settings and select GPU.
2. No need to understand all the details !