Transfer Learning

Practical II - Machine Learning

1 Objectives

The goal of this practical is to practice the use of Convolutional Neural Networks (CNNs) and Transfer Learning. In this practical you will compare three strategies for training CNNs, two of which use Transfer Learning, in an image classification problem. The idea is to study how the weights learned by a model trained for one task can be used in a different task. This practice is very common, especially considering how much time it takes to train Deep Learning models, even a few days or weeks in some cases.

2 Task

In this practical you will work with an image dataset called CIFAR-10[1]. This dataset has 60000 images of 32x32 pixels, of which 50000 are for training and 10000 for test. Each image belongs to one of 10 classes, numbered from 0 to 9. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The dataset can be easily downloaded using Keras with the following Python commands:

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

The first step is to separate the dataset in two parts A and B. In part B you must select all images belonging to classes number x and y, where x and y are the last two digits of you university registration number. If the last two digits are the same, go back one digit from x until both digits are different. For example, if your registration number is 2020387499, part B will have the classes of number 4 and 9.

Once this is done, you will need to train a CNN to classify the images from part A using the Keras API for TensorFlow. The CNN has to have four convolutional layers and two Max Pooling layers (not necessarily in this order) and two Fully Connected layers at the end. The activation function ReLU must be used in every layer that has activation, except the last one which must use Softmax activation. It will also be necessary to use Dropout layers to regularise the model. You will need to adjust the hyperparameters of the network so that the value of the loss function is minimised during training and to control overfitting. To measure the generalisation of the model, use only the test set (it is not necessary to use cross validation) and the accuracy. The model must be trained for 20 epochs.

After training the model using the data from part A, you must test the three strategies listed below to train a model for the data from part B. You must use the same architecture of the CNN previously trained, and also train for 20 epochs. The accuracy must be used to evaluate all the models.

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`

1. **No Transfer Learning**: Train the network from scratch, initialising the weights randomly.
2. **Fine-tuning in one layer**: Use the weights of the network trained for part A and fine-tuning in the last Fully Connected layer.
3. **Fine-tuning in two layers**: Use the weights of the network trained for part A and fine-tuning in both Fully Connected layers.

At the end you must compare the results for the three strategies, conclude which one achieved the best test accuracy and explain what was the effect of using Transfer Learning. For each of the four networks (one for part A and three for part B) show, in one plot for each, the convergence of the loss function and the training and test accuracy.

It is recommended to use a GPU to train the CNN models to accelerate the training. If you do not have access to a GPU with Cuda, you can use Google Colab (`https://colab.research.google.com/`). This tool allows running a Notebook in the cloud using a GPU. Simply create a notebook (which is saved in Google Drive) and in the *Runtime* menu, in the option *Change Runtime type*, select GPU as the *Hardware Accelerator*.

3 Submission

The practical needs to submitted as a Jupyter (Ipython) notebook. The notebook has to include all the code (with appropriate comments) necessary to run all the experiments, present the results using text, plots and tables, the explanations of what is being done and the interpretation of the results. Only the notebook is to be submitted. The clarity and organisation of the notebook will also be evaluated. The teaching assistant has to be able to reproduce all the experiments simply by running all the notebook cells in order.

The notebook has be submitted to Moodle by 02/07 at 23:59 (only the .ipynb file has to be submitted). **Practicals submitted after the deadline will not evaluated**. **There will be no deadline extensions**. **In case of plagiarism, all students involved will be graded zero and reported to the CS department**.

4 Assessment Criteria

- Correct implementation of the networks, the experiments and model evaluation (50%)
- Clear, concise and non-ambiguous presentation of the experiments and results. (20%)
- Convergence of the models (the training error is minimised and the model is regularised) (20%)
- Organisation of the information presented in the notebook. (10%)

Note that you will not be evaluated based on the accuracy obtained in the experiments, only the process itself.