M 124 Coursework 4

Y. Panagakis and **K. Panousis**

Due Date: June 8, 2021, 23:59

1 Introduction

For the fourth and final coursework, we focus on Neural Networks.

In the last few lectures, we've explored Deep Neural Networks and their convolutional variants. We started from the simple perceptron, a milestone in learning theory and showed how we can construct and train multilayer structures for either classification or regression. We examined the difficulties of training deep nets and introduced some major approaches for regularizing the model, including Dropout and Batch Normalization.

In this coursework, you will delve deeper into the implementation of Deep Neural Networks and explore some core choices that you must consider when approaching a specific task (dataset/application). These include but are not limited to the network architecture, the optimization process and the training hyperarameters. Thus, in the following you are asked to explore the effect of these choices in a widely used dataset, namely CIFAR-10.

2 Neural Networks **(100 Points)**

In this task, you will implement both standard feedforward neural networks as well as convolutional ones. To this end, you need to create two separate files:

- nn.ipynb/nn.py: Train a fully connected network.
- cnn.ipynb/cnn.py: Train a convolutional neural network.

Each file should implement the necessary functionality in order to train and evaluate the performance of a considered model (either fully connected or convolutional DNN), including a training and a testing function.

Use the built-in functions of pyTorch to load the datasets as we saw in the respective tutorials and for the rest of the training and testing processes. For a faster training/evaluation do not forget to use the GPU runtime in Colab.

2.1 Training and Evaluation **(30 Points)**

Train a regular NN and a CNN with SGD using the default hyperparameters and a batch size of 64 and for a maximum of 100 epochs.

Specifically, consider a 4-layer fully connected network, where each layer will comprise a number of neurons of your choice and a LeNet-5 like architecture for the CNN variant. Recall that a LeNet-5 network

comprises 2 convolutional layers and 3 FC layers and there exist different variants for the number of filters, the size of the kernel matrix, e.t.c. Report your architectural choices in the final deliverable.

- Examine the statistics and plots of training and validation error (generalization).
- How does the network's performance differ on the training set versus the validation set during learning?
- Show the plots of error curves for both networks.
- How do the two networks fare against each other?

2.2 Optimization (30 Points)

In this part, you'll explore the effect of your choices concerning the optimization procedure.

- Try 2 different settings of the learning rate from 0.001 to 1.0. What happens with the convergence of the algorithm? Look at both final values of the cross entropy loss and the classification accuracy of the network. Produce the corresponding plots showing the changes in these values per epoch.
- Using the "best" learning rate from the previous task, try 2 different values of the momentum from 0.1 to 0.9 (the default value in your previous tasks was 0). Does the momentum affect the convergence rate? If so, how big is the difference and why do you think this happens?
- How would you choose the best value of these parameters?

2.3 Model Architecture (30 points)

- Fix the momentum of the optimizer to be 0.9.
  - Try 2 additional different values of the number of hidden units for each layer of the fully connected network (in the range 200-500), and 2 values for the number of filters for each layer of the Convolutional Network (in the range 20-128), e.g. 64-128, 32-64, 64-64 or any other combination.
  - You might need to adjust the learning rate and the number of epochs for each considered architecture if you observe divergent behavior in your training process.
  - Comment on the effect of the network size on the convergence properties and generalization of the network.

2.4 FC Networks vs CNNs (20 points)

- Calculate the number of parameters (including biases) in each type of network.
- Compare the performance of a convolutional net and a regular network for a similar number of parameters (To avoid the complexity of re-training FC and CNN models, you can choose similar number of parameters in one of your previous explorations, e.g. in Task 2.3).
  - Which one leads to better generalization and why?
  - Plot the first layer outputs (for 3-4 filters) of the CNN. Briefly comments on these visualizations.

General Notes

- In this coursework, you are asked to approach a standard task in Image Classification using Deep Neural Networks. As was the case in the previous courseworks, I highly recommend to take your time and understand the fundamental techniques and rationale behind the introduced approaches and avoid searching ready-made solutions online.
- Individual Coursework.
- Must be submitted via e-class. E-mailed submissions will not be accepted.
- Programming:
  - For the programming part, you can turn to StackOverflow if you bump into some kind of bug or problem, but again I strongly advise against copying code for these simple tasks.
  - As mentioned in the first tutorial, it is best practice to use Virtual Environments for your projects to avoid breaking the system. You can use either Virtual Environments of native python venv or install an Anaconda/Miniconda platform. Or simply use Google Colab.
  - Implement your approach using a Jupyter Notebook, with sufficient but not redundant comments.
- Typesetting:
  - I would suggest using LaTeXfor reporting the results. LaTeXis a very powerful typesetting system, most commonly employed in research papers and reports. It provides all the necessary tools for easily typesetting equations, arrays and more, while avoiding the hassles of other editors. This CS was written in LaTeX. You can use Overleaf for online editing.
  - A template will be provided.