Image Interpretation – Assignment 5

This assignment on **Deep Convolutional Neural Networks** covers the deep networks labs from the 7th and the 14th of November, and is graded by a maximum of 9 points.

In order to submit the results, send a ZIP file with your implemented code (functions prefixed with `ii_` and any "helper" functions you wrote) to

`<riccardo.delutio@geod.baug.ethz.ch>`

with subject

`Image Interpretation 2019 Assignment 5`

no later than on the

`20st of November, 2019.`

In addition to the code, include a report (**max. 8 pages** PDF) explaining the structure of the code and the `python`/MATLAB functions used. This includes the reasons for choosing particular functions as well as a short justification of their parameter setting. For the more complicated tasks, the choice of the underlying data structures and algorithms should be explained too. We encourage you to add also diagrams, illustrations, and figures into the report when appropriate, but it is not necessary to copy the related theory from the lecture slides. The report should not contain any code snippets (but the code should contain comments if appropriate).

Team work is not allowed. Everybody implements his/her own code and writes his/her own report. Discussing issues with others is fine, sharing code and/or report with others is **not**. If you use any code fragments found on the Internet, make sure you reference them properly.

Exercise 1. 2 P.

In this first exercise we are going to get an idea on how simple neural networks behave for simple classification tasks. We'll make use of two different javascript based neural network to train and modify an existing model. Let's start with a very simple one, a simple neural network with only fully connected layers and without convolutions. Open a web browser on the page `http://playground.tensorflow.org/` and start playing with the interface. For each dataset set the noise level to 30. Click play on the top-left to start training. To assess the quality of the network look at the values train and test loss at the top right.
Tip: Sometimes it's easier to see the result if you discretise the graph.

1. For each dataset (the four boxes under DATA) explore the best features that make the network converge faster.
2. For the spiral dataset and default network how does the batch size (bottom left slider) influence the results? How does it relate with the learning rate (central top)?
3. Try all the different activation layers (top slider), what are their differences? Why?
4. How does the number of hidden layers and the number of neuron per layer affect the results in terms of quality (test loss) and convergence speed?
5. Compare the performance (convergence rate and accuracy) of a network for regression and classification? Explain the differences.
6. Play with different types and rates of regularization (top slider) and explain the changes you observe and the influence on the results.

Explain each question and motivate it with a screenshot of the experiment.

The following part is optional, but fun! Open this page `https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html` and try to tweak all the parameters you can to obtain the best network (the highest validation accuracy). Scroll down the page and peek inside the network and have a look at different activation and weight as the network gets trained (click pause to make it freeze). Check how the prediction works on the test set in the last part of the page.

Exercise 2. 3 P.

In this exercise you have to train a new Convolutional Neural Network from scratch for the classification of images. We are going to make use of Matlab own deep learning toolbox.

1. The aim is to achieve 99% accuracy (on validation set) for the MNIST dataset `http://yann.lecun.com/exdb/mnist/`. You can find the script to train a simple CNN on the MNIST dataset in `nn_script.m`.
2. You are allowed to do whatever you want (except copy pasting) with the network as long as it is explained in your report. Feel free to change the architecture of the network as well as parameters (e.g. learning rate, kernel sizes, ...). You can try to guess parameters manually of you want, just make sure that it performs better than 99% on the validation set (at least 10% of the dataset).
3. Sketch the final network architecture in your report.
4. Make sure you train the network on the GPU, otherwise it will be too slow.
5. Explain the plot produced by the program.

Exercise 3. 4 P.

There is a competition `https://www.kaggle.com/c/dogs-vs-cats-iminit-2019`. You should have at least 75% accuracy on the public leaderboard to get basic points on this exercise. The first 3 places will have extra points (5 for the first place, and 4 for the second and the third place). Use all your knowledge and experience to attack the problem.