Artificial Intelligence
Wintersemester 17/18
Prof. Björn Ommer
ommer@uni-heidelberg.de

Exercise sheet 08: Exercise 08

Due on *22.12.2017, 2 pm*, 2pm.

Question 1: Classification of Fashion-MNIST dataset (6P)

We continue to work on the CNN-example from the last exercise sheet (question 4). In the lecture we have considered several techniques to improve performance of a neural network and here we want to test some of them.
Before starting let switch to “more interesting” dataset: Fashion-MNIST[1]. It contains images of 10 different types of clothes and shoes (e.g., pullovers, dresses, sneakers) but in low-resolution. It has exactly the same amount of data as classical MNIST and the same number of classes. That means you can easy replace one by another without changing too much.
In all tasks below we use the network architecture from MNIST-example[2]. Report the results on **the test set** after **10** epochs of training.

1. *Pre-processing*: please use or define a new **torchvision dataset** to download and load the data. Use the **DataLoader class** to generate batches for training and testing. Visualize several images from the dataset with the corresponding labels (e.g., "coat", "sandals", etc).

2. *baseline* (1P): consider the network from the MNIST-example[2]. How many parameters has this network? Report a number of parameters of each layer together with the total amount of parameters. Train the network on Fashion-MNIST dataset. Report the accuracy of the best epoch.
   Hint: you may want to decrease the learning rate to make the network learn.

3. *data augmentation* (1P): utilize several data-augmentation techniques (e.g., random horizontal flip, random cropping, etc). You may check what **torchvision** offers. Visualize the effect of each transformation on a sample from the dataset. Train the model again from scratch and report the accuracy of the best epoch.

4. *batch normalization* (1P): add batch normalization in the network and train it from scratch. Report the accuracy of the best epoch.

5. *finetuning* (2P): train the model on MNIST and save the weights of the best epoch. Use this weight to initialize the network when training on Fashion-MNIST. Moreover freeze all weights except of the last fully-connected layer

[1] `https://github.com/zalandoresearch/fashion-mnist`
[2] `https://github.com/pytorch/examples/blob/master/mnist/main.py`

and train the network for 10 as in all other cases. Report the accuracy of the best epoch.

6. *visualize the learning curves* (1P): visualize how the loss and accuracy of the network is changing over the time **for each of 4 above cases**. Make two plots: for training and test sets correspondingly. What do you see? Which method worked best for you?

Question 2: t-SNE visualization (1P)

t-SNE[3] is a very useful dimensionality reduction method. One can seen an example of a t-SNE embedding on the web-page of Fashion-MNIST: it was used to create a 3D visualization of the dataset. Your task is to apply t-SNE to map samples from high-dimensional feature space of fc1-layer of the network we used so far onto a plane. For this use an arbitrary model from above, obtain fc1-feature vectors for all images in training set and fit a transformation, that maps this features into Cartesian coordinates. You may use the t-SNE implementation provided in *sklearn python package*. Please visualize the obtained points. Use different colors for different classes. What do you observe? How good is the fc1-features for the further classification?

Question 3: What the network did learn? (3P)

In this task we want to investigate what is important for a network to predict a certain class. One way of doing this is to generate an image that highly activates a neuron responsible for predicting one specific class. The algorithm behind this is called *activation maximization* (for more information and examples see `https://distill.pub/2017/feature-visualization/`). A simpler version of this algorithm is summarized below. Your task is to implement it and to apply to one of the networks from question 1. In the assignment folder you can find a help file *dreamer.py*, where you are should fill missing parts.

- start from an empty image
- select a class
- apply *gradient ascent* to maximize the activation of the specific neuron in the last fc-layer.

[3] `https://lvdmaaten.github.io/tsne/`

The output of this method is not always easy to recognize. We advise to consider easier classes like "T-shirt/top", "Trouser", "Ankle boot". To give you a feeling, what one can expect we show below our results. Plot a confusion matrix of the results on the test-set. Does the accuracy on different classes correlate with the quality of generated images?

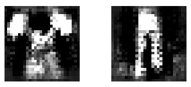

Table 1: Our results on "T-Shirt/top" and "Trousers".

Note: Submit exactly one ZIP file and one PDF file via Moodle before the deadline. The ZIP file should contain your executable code. Make sure that it runs on different operating systems and use relative paths. Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. The PDF file should contain your written code, all figures, explanations and answers to questions. Make sure that plots have informative axis labels, legends and captions.