Assignment 4: Convolutional Neural Networks
MA-INF 2313: Deep Learning for Visual Recognition

Due Date Theoretical: 08.11.2017
Due Date Programming: 15.11.2017
Assistant: Fadime Sener `sener@cs.uni-bonn.de`

1 Theoretical Exercises (*15 pts)*

1. *(7 pts) Backpropagation through Convolutional Layer :*

   The output (before the nonlinearity) of a convolutional layer with weights $w \in \mathbb{R}^{k \times k}$ with respect to input image $x \in \mathbb{R}^{d \times d}$ is given by the convolution $y = x * k$. Note that for now, we consider only a single input and output channel.

   Derive the quantities $\frac{\delta y}{\delta w}$ and $\frac{\delta y}{\delta x}$.

2. *(8 pts) Adding constraints on the weights of a CNN:*

   Consider a neural network, such as the convolutional network or a general feed-forward network, where each unit computes a weighted sum of its inputs of the form

   $$a_j = \sum_i w_{ji} z_i,$$

   where z_i is the activation of a unit, or input, that sends a connection to unit j, and w_{ji} is the weight associated with that connection. Let h be a nonlinear activation function, then $z_i = h(a_i)$.

   We now modify this neural network such that multiple weights are constrained to have the same value. Discuss how the standard backpropagation algorithm must be modified in order to ensure that such constraints are satisfied when evaluating the derivatives of an error function with respect to the adjustable parameters in the network.

2 Programming Exercises : TensorFlow (*15 pts)*

In this programming exercise, you will set up and train a CNN to identify and distinguish between faces of 20 different people. The dataset is a subset of the **ORL Database of Faces** [1] and is provided in the file `ORL_faces.npz.zip`. The size of each image is 92×112 pixels, with 256 grey levels per pixel. It is recommended to visualize the dataset as a sanity

[1] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

check. You can use `load_ORL_faces.py` to load the data. Use the given training (trainX, trainY) and test (testX, testY) splits.

1. *(8 pts)* Code your own CNN with at least two convolutional layers. Train your CNN on the provided training set and report your classification accuracy on the provided test set. Assume that each convolutional layer consists of a convolution operation followed by a relu and max-pooling.

   (a) *(2 pts)* Sketch the implemented network architecture on a piece of paper and attach it with the solutions for the theoretical part. Report the number of parameters and neurons in your network.

   (b) *(3 pts)* Train your CNN using different learning rates. For each learning rate plot a curve showing how the classification error decreases over each iteration. Fix the best learning rate for the next steps.

   (c) *(3 pts)* Plot the training loss over the iterations and report the final test accuracy.

2. *(4 pts)* Add regularization to your CNN in the form of dropout layers. Train and test your network on the specified training and test set. Plot the training loss over the iterations and report the final test accuracy.

3. *(3 pts)* Visualize the convolutional filters of the first layer in the trained networks.