University of Salzburg — Lecturer: Roland Kwitt

Computer Vision – Proseminar (911.909)

Exercise sheet F (Jan. 23, 2017) — Hand-in by **Feb. 3, 2017**

Training a CNN on CIFAR-10

Exercise 1. 20 P.

The task is to design, train and test your own little CNN to classify **CIFAR-10** images. The CIFAR-10 dataset contains 60000 labeled images from 10 classes (size 32x32 pixel).

You can start from the `TORCH` blog entry on training CNNs on CIFAR-10 – click here. The blog links to code on GitHub that you can (and should) use.

Currently, the network models that come along with the code are (1) a VGG-style network (see Simonyan Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv:1409.1556) and (2) a *Network-in-Network (NiN)* example (see Lin et al., *Network In Network*, arXiv:1312.4400). Read the original papers and compare the descriptions in the papers to their actual implementations (under the `models` directory).

I suggest to go through the **following steps** to complete this exercise:

1. Clone the code from the GitHub repository and try to understand its structure, i.e., how data is loaded, how training & testing is accomplished, etc. – **Read the TORCH blog entry!**
2. Start training the NiN network on your CPU (or GPU) for a couple of epochs (e.g., 50).
3. Create a file `tiny.lua` and **define your own network** in that file. Make sure you document each design choice. Below is a short code sample that you can use to test if your network design produces the desired output (in terms of output size – remember we have a 10-dim. classification problem). You would just have to replace `nin.lua` with `tiny.lua`.
4. Train your network for a reasonable number of epochs (on your CPU, or GPU if you have one). Choose a network size that allows you to run through at least 50 epochs in reasonable time (e.g., a couple of hours).
5. Hand-in the network definition + a figure (e.g., in the for of a PNG, or a PDF file) that plots the training and testing error over the number of epochs that you trained.

```
require 'nn'

torch.setdefaulttensortype('torch.FloatTensor')

local model = nn.Sequential()  -- create a Sequential container
model:add(dofile('nin.lua'))   -- add the NiN network

input = torch.Tensor(5,3,32,32) -- random input
output = model:forward(input)   -- network output (forward pass)
```

Total #points: 20 P.