Deep Learning	October 25, 2015
Exercise 0	
Lecturer:Lior Wolf	*TA: Adam Polyak*

Introduction

The purpose of this exercise is to familiarize you with Torch7. This exercise will focus on Lua and simple torch usage. You are only required to submit questions under "Handin" section, the rest of the questions are for self-learning. It is important to go over them, since we will not cover basic Lua/Torch syntax during the course.

Enviroment

During the exercises we will use Torch7 which is a "matlab like" environment for deep learning. You have a couple of options:

- If you have a Ubuntu(12+) or Mac, you can install it from http://torch.ch. The installation process is simple and straightforward.
- Connect to nova, and then run the following:
  `/usr/local/lib/torch/bin/qlua -e "repl = require 'trepl';repl()"`
  Do not forget to enable X11 forwarding (details here).

Introduction to Lua and Torch

Question 0: Lua

It's time to learn Lua, enter the following link: http://tylerneylon.com/a/learn-lua/. It contains several code snippets that explain basic features of the lua language, go over it. Much more about lua here.

Try to answer the following guidelines questions:

- Why is the `local` keyword important? (hint: default variable scope is not local)

- What is the difference between a.f() and a:f()? (hint: one implicitly adds self as the first argument)
- What does require do, and why do we sometimes capture its return value but sometimes not? (this is a way to isolate things into namespaces to prevent naming conflicts, related to the answer to why we need to use local)
- What is the Lua equivalent of a list object? of a dictionary? How do you iterate over each of these?

Question 1: Torch

Go over the Torch tutorial here: http://torch.ch/docs/five-simple-examples.html. Every time you see "luarocks install ..." skip it - all the packages were installed.

Torch uses Tensor, which is a generalization of 1-d vectors and 2-d matrices to n dimensions. Go over the various features available:

- https://github.com/torch/torch7/blob/master/doc/tensor.md
- https://github.com/torch/torch7/blob/master/doc/maths.md

Notice how some operations can be done "in-place" while other are not. Specifically, think about the difference between:

```
th> t = torch.Tensor(4,4)
th> t:add(-4)
```

And,

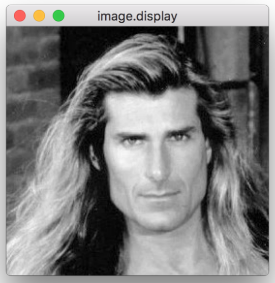
```
th> t = torch.Tensor(4,4)
th> t = t - 4
```

Question 2: Image display

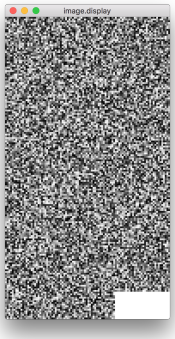
In the previous question, you displayed plots. Torch also allows you to display images, go over the following:

```
-- for an RGB image
th> require 'image'
th> image.lena()
th> image.display(l)
```


```
-- for gray scale image
th> fabio = image.fabio()
th> image.display(fabio)
```




```
-- weights of a neural network can also be displayed
th> require 'nn'
th> net = nn.SpatialConvolution(1, 64, 16, 16)
th> image.display{image=net.weight,zoom=3}
```




```
-- and of course the results of running a neural network
th> n = nn.SpatialConvolution(1,16,12,12)
th> res = n:forward(image.rgb2y(image.lena()))
th> image.display{image=res:view(16,1,501,501), zoom=0.5}
```

Handin

Question 3: Channel normalization

Write Torch code to normalize MNIST dataset globally. In other words, reduce the mean over all pixels and all samples from each sample. Same goes for variance. Try to make the code efficient as possible (memory allocation wise).

To download the data use:

```
wget https://s3.amazonaws.com/torch7/data/mnist.t7.tgz
tar xvf mnist.t7.tgz
```

Then load the data like this:

```
th> train = torch.load('mnist.t7/train_32x32.t7', 'ascii')
th> test = torch.load('mnist.t7/test_32x32.t7', 'ascii')
th> train
{[data]   = ByteTensor - size: 60000x1x32x32
 [labels] = ByteTensor - size: 60000}
```