STATISTICAL MACHINE LEARNING (SML2016)
3. COMPUTER LAB
Transfer Learning for Convolutional Neural Networks

Jan Drchal

1 Overview

The goal of this laboratory lab is to show how transfer learning can be used to allow or at least ease solving a classification task when not enough training data are available but we already possess a classifier for a similar problem.

Your task can be summarized by the following steps:

1. Work with the well known MNIST dataset [1]. MNIST is a database of handwritten digits (grayscale images of 28×28 pixels) containing $60,000$ training and $10,000$ test examples. See Figure 1 for an example.

2. Train a Convolutional Neural Network (CNN) with softmax output layer using only the data for digits 0 to 5. Network architecture is depicted in Figure 1.

3. Train a series of CNNs classifying digits 6 to 9 using only a proportion P of available data. The architecture of all CNNs will be the same as in the previous step with an exception of the number of output units (4 instead of 6).

4. Train a series of CNNs using the same architecture and train/test sets as in the step 3 but fix all convolutional layer weights to those obtained by training the CNN from the step 2 (this is where the *transfer learning* actually happens).

5. Compare the convergence of the networks from the steps 3 and 4.

2 Download

The data can be downloaded from this link:

`https://cw.fel.cvut.cz/wiki/_media/courses/be4m33ssu/ann_data.zip`

[1] `http://yann.lecun.com/exdb/mnist/`

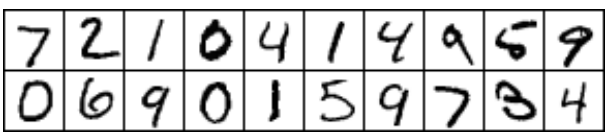

Figure 1: **MNIST digits sample.**

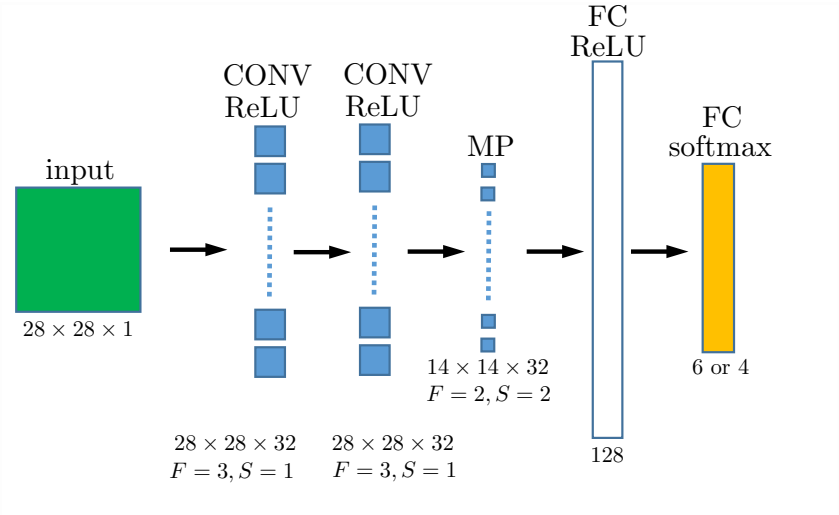

Figure 2: **CNN architecture.** Linear layers preceding ReLU and softmax layers not shown.

The zip file contains:

`snippets.py` Python code to import the dataset as well as some Keras basics. It also shows how the dataset was exported for Matlab users. Note that the sizes of the training set for the digits 6 to 9 are adjusted to correspond with the recommended batch size of 100.

`mnist.mat` Datasets exported for the Matlab users. The file contains the following arrays:

`X_trn05`	digits 0 to 5 train set inputs
`t_trn05`	digits 0 to 5 train set targets
`X_tst05`	digits 0 to 5 test set inputs
`t_tst05`	digits 0 to 5 test set targets
`X_trn69_P`	digits 6 to 9 train set inputs, where `P` is the proportion
`t_trn69_P`	digits 6 to 9 train set targets, where `P` is the proportion
`X_tst69`	digits 6 to 9 test set inputs
`t_tst69`	digits 6 to 9 test set targets

3 Task assignment

Assignment 1 (7 points) *Perform all steps described in the Section 1 for the dataset proportions* $P \in \{0.005, 0.01, 0.1, 1.0\}$ *which correspond to the train set sizes* 100, 200, 2300 *and* 23900. *The test set size is* 3969. *Each configuration should be run for at least three times. Use the following architecture of the network:*

1. *convolutional layer of 32 filters* (3×3 *kernels, stride* 1*), apply ReLU activation function,*
2. *convolutional layer of 32 filters* (3×3 *kernels, stride* 1*), apply ReLU activation function,*
3. *max pooling layer* (2×2 *kernel),*
4. *fully connected layer of* 128 *ReLU units,*
5. *fully connected layer with a softmax classifier.*

Record an accuracy measured on a test set every 1000 *samples processed by a learning algorithm of your choice. The report should contain:*

- *An average test set accuracies achieved for all* 6 *to* 9 *CNNs and their transfer learning versions for all proportions* P*. Give tables as well as figures. You can use box plots to visualize the accuracy distributions over the multiple runs per configuration.*
- *Visualization of a progress of average test accuracies during learning for both types of* 6 *to* 9 *CNNs for all proportions.*
- *A short discussion of the results.*

Remark: You should have no problem in achieving accuracies over 99% *for both* 0 *to* 5 *and* 6 *to* 9 *networks (for proportion* $P = 1$*).*

Assignment 2 (3 points) *Perform the same set of experiments as given by Assignment 1. Initialize the weights of the* 6 *to* 9 *CNNs using* 0 *to* 5 *CNN but do not fix them during the learning phase. Add a short discussion of the results to the report.*

Assignment 3 (3 points) *Perform the same set of experiments as given by Assignment 1. Modify all networks to use dropout for the layers* 4 *and* 5 *(use dropout probability 0.5). Add a short discussion on what changes when dropout is involved.*

Remark: You don't have to implement Convolutional Neural Networks and their learning as there are many existing frameworks available [2]*. For Python we suggest Keras* [3] *for which code snippets are added to the archive. Matlab users might use Caffe bindigs* [4]*.*
Hint: The experiments might be computationally expensive so use GPU if possible and more importantly start as soon as possible.

References

[2] `http://deeplearning.net/software_links/`
[3] `https://keras.io/`
[4] `http://caffe.berkeleyvision.org/tutorial/interfaces.html`