GMDL, HW #5

Oren Freifeld and Ron Shapira Weber
The Department of Computer Science, Ben-Gurion University

Release Date: 26.05.22
Submission Deadline: 01.07.22, 23:59

Abstract

This assignment focuses on fully connected (FC) neural networks, convolutional neural networks (CNNs) and Transfer Learning.

Contents

Version Log

- 1.0.0, 26.05.22 Initial release.

Contents

1 Preliminaries

Read before you start

- In this assignment we will use PyTorch to build and train simple neural networks.
- We recommend working with a Colab notebook. This will save you the need of local installations on your own machine.

- What is Colaboratory? Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with zero configuration required, free access to GPUs (use this option when you train your neural network – it would run much faster. Pay attention though: GPU time is limited) and easy sharing.
- After you finish the assignment, add Ron Shapira Weber's email address (`ronsha@post.bgu.ac.il`) to your colab notebook. In addition, download the notebook as an .ipynb file and submit it, together with its already-run-and-exported-to-pdf version (file→ print→ use 'save as PDF' as destination printer), in Moodle as usual.
- Some tutorials for working with PyTorch are available at `https://pytorch.org/tutorials/`. You are encouraged to use any function or module you see fit. Most of what you need in this assignment is already implemented in PyTorch, and there is no need to implement it yourself.
- It is recommended (but not mandatory) to use the TensorBoard to plot the loss and accuracy of the models in this assignment. You can see **how to use TensorBoard with Pytorch** here.

2 Data

For this assignment we will use the Street View House Numbers (SVHN) Dataset. This dataset contains real-world images of digits and numbers in natural scenes (signs of house numbers). For more information see `http://ufldl.stanford.edu/housenumbers/`. Note that this dataset has two formats: Full Numbers and Cropped Digits. We will use the Cropped Digits version, which is also what you will get from `torchvision.dataset` - Check out this documentation for help: torchvision datasets.

Remark 1 ***Validation Set*** *- A validation dataset is a dataset of examples used to tune the hyper-parameters (*e.g.*, the architecture) of a classifier. It is sometimes also called the development set or the "dev set". An example of hyper-parameters for artificial neural networks includes the number of hidden units in each layer. It, as well as the testing set (as mentioned above), should follow the same probability distribution as the training dataset (see Wiki).* ◊

Computer Exercise 1 *Load data using PyTorch.*

(a) Use `torchvision.dataset` *to download and load the SVHN data. Note that you need to load both the train and test data.*

(b) Split ***the train data*** *into train (*80%*) and validation (*20%*). (*i.e.*, take 20% of the train data for validation, and do not use the validation set for training). You can use* `torch.utils.data.random_split`*.*
Tip: for debugging you can use `random.seed()`*.*

(c) Create a dataloader for train, validation and test data. You will use this dataloader in the training phase to get data batches.

(d) Visualize several images in order to become familiar with the dataset. ◊

3 Baseline networks

The task we aim to solve is classification. The input to our neural net will be an image and the output will be a discrete label. As our dataset contains images of digits, labeled 0–9, think what size the last layer of any neural net that you will train to solve this task should be.

Computer Exercise 2 *Implement two neural network architectures: a fully-connected neural network (FC); a convolutional neural network (CNN). The specific architecture parameters of the networks are specified below.* ◊

3.1 Fully-connected neural networks architecture

In this network, all the layers will be fully-connected; *i.e.*, each neuron in a certain layer is connected to all the neurons in the previous one. The first layer is (and always will be) the input layer, followed by:

Layer 1: 128 neurons, followed by a ReLu activation.

Layer 2: 64 neurons, followed by a ReLu activation.

Layer 3: 10 neurons (as the number of distinct labels).

3.2 Convolutional neural networks architecture

Our CNN will consist of:

Layer 1: conv layer with kernel size of 3, stride of 1, padding of 1, with 10 output channels, followed by a ReLu activation.

Layer 2: max pooling layer with kernel size of 2, stride of 2 and zero padding.

Layer 3: conv layer with kernel size of 3, stride of 1, padding of 1, with 20 output channels, followed by a ReLu activation.

Layer 4: max pooling layer with kernel size of 2, stride of 2 and zero padding.

Layer 5: a fully-connected layer of 64 neurons, followed by a ReLu activation.

Layer 6: a fully-connected layer of 10 neurons (as the number of distinct labels).

4 Training

Computer Exercise 3 *In this section, train the two neural networks you build. Use the following parameters for the training of both networks:*

- *Cross-Entropy loss*
- *Adam optimizer with a 0.001 learning rate*
- *30 epochs* ◊

You can use the following template for the training:

```
# epochs loop
      # batches loop
            # forward pass
            # calculate loss
            # back propagation to compute to gradients
            # updates the weights
```

Computer Exercise 4 *Both of the above NN has hyper-parameters defining the architecture. For the FC net try different sizes for the hidden layers, and for the CNN try different kernel sizes. Use the validation dataset to compare between the different options.*
For the FC try the following layer sizes:

- *512 and 256*
- *64 and 32*

For the CNN try the following kernel size (for the conv layers):

- *10*
- *1*

Compare the three options for each net:

- *Plot the loss per epoch for the training and validation sets for each architecture option.*
- *Plot the train and validation accuracy for each architecture option.* ◊

You can use any logging platform of your choice or manually plot the graphs. If you choose the second options make sure the plots are clear for us to read , think in what graph you can locate overfitting?

Problem 1 *report your findings, which of the architectures preforms best on the validation set? Based on the validation set, is the net overfitting? underfitting?*◊

5 Evaluations

Computer Exercise 5 *compute the test accuracy for the model that preformed best on the validation set.* ◊

Problem 2 *Is it guaranteed that the model you chose based on the validation set will give the best performance on the test set? What basic assumption do we make when using the validation to choose the hyper-parameters values?* ◊

6 Transfer learning

In this section you will design a new network using transfer learning from ResNet-18 pre-trained over Image-Net(as explained in the ps and can be seen in this link). The goal is to outperform the basic nets you implemented earlier.

Computer Exercise 6 *Choose* ***one*** *of the options to transfer learning we learned about:*

- *Finetuning ResNet*
- *ResNet as a fixed feature extractor*

Train the model to solve the SVHN classification task. Use the same parameters for the training as in the previous two networks. ◊

Problem 3 *Specify which transfer learning method you chose and explain why.*◊

Problem 4 *Compare the results of the new network with both CNN and FC networks. Does it improve performance? Provide a brief hypothesis on why the increase/decrease in performance occurred.* ◊

Problem 5 *Experiment with different learning rates. Demonstrate the effect of the value of the learning rate by plotting the loss during training and and explain the results.* ◊