# CV202, HW #8

Oren Freifeld and Meitar Ronen
The Department of Computer Science, Ben-Gurion University

Release Date: 26/5/2020
Submission Deadline: 7/8/2020, 20:00

**Abstract**

Please make sure you have carefully read the general instructions in the course website at https://www.cs.bgu.ac.il/~cv202/HW_Instructions. These instructions address, among other things, what to do and not do with figures, Jupyter Notebooks, *etc*., as well as what compression files are legit or not.

The purpose of this assignment is to get you acquainted with deep learning and gain some hands-on experience using PyTorch, a popular and easy-to-use framework. In case you experience technical issues with your personal computer, you can use Google Colab, which is a free framework to execute Jupyter notebooks on Google's cloud servers (see https://colab.research.google.com/notebooks/intro.ipynb#). In this assignment you will implement three neural networks: two baseline networks and one which you will design with the goal of outperforming the baselines. You will need to submit (in a single .zip file):

1. Python code implementing all the required sections below.
2. Images of the required plots in 2.4 problems 1,2 and 5.
3. Answers to problems 1 - 7.

## Version Log

- 1.00, 26/5/2020. Initial release.

## Contents

# 1 Preliminaries

In this assignment we will use PyTorch https://pytorch.org/ to create and train neural networks. To work with PyTorch, you will need to install the torch and torchvision packages.
Using conda:

```
conda install pytorch torchvision cudatoolkit=10.1 -c pytorch
```

See also the installation instructions here, https://pytorch.org/get-started/locally/.

Some tutorials for working with PyTorch are available here https://pytorch.org/tutorials/.

You are encourages to use **all** of the functions that are implemented in PyTorch. Most of the operations in this assignments are already implemented, and there is no need to implement it yourselves.

# 2 Data

For this assignment we will use The Street View House Numbers (SVHN) Dataset. This dataset contains real-world images of digits and number in natural scenes (house numbers). For more information see http://ufldl.stanford.edu/housenumbers/.

Part (i) use `torchvision.dataset` to download and load SVHN data. Note that you need to get both train and test data.

Part (ii) split the data into train (80%) and validation (20%). You can use `torch.utils.data.random_split`.
*tip: for debugging you can use* `random.seed()`.

Part (iii) create a dataloader for train, validation and test data with a batch size of 64. You will use this dataloader in the training phase to get data batches.

Part (iv) visualize the first batch of images

# 3 Baseline networks

In this part you will implement two neural networks, a fully-connected one (FC) and a convolutional one (CNN).

## 3.1 First baseline

In this network, all the layers will be fully-connected, meaning, 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 Second baseline

This network will be a convolution (conv) neural network comprised 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

In this section we will train the two baselines using:

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

The template for the training should look like:

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

## 5 Evaluations

In this section you will evaluate the two networks performance by plotting the networks' loss on the data. For this, it is recommended (but not mandatory) to use the *TensorBoard* package (https://www.tensorflow.org/tensorboard/get_started).

**Problem 1** *For each network, plot the loss during the training phase.* ♢

**Problem 2** *For each network, plot and compare the accuracy of the network on the training set with the accuracy of the validation set and the test set. Which of the networks have overfitted?* ♢

# 6 Design your NN

In this section you will design your own network, with the goal of outperforming the two baselines. You can choose the network's architecture (number and types of layers, activation functions, etc.). Also, incorporate **at least** one of the following options:

- Data augmentation
- Spatial Transformer Network
- Dropout
- Regularization

  Feel free to use more additions to your network besides the above.

**Problem 3** *Train and evaluate (as specified at section 5 your network with and without the above addition(s).* ◊

**Problem 4** *Why did you choose these options? Give a* ***brief*** *explanation of how they work.* ◊

**Problem 5** *Compare your network with and without that addition. Does it improve performance? Provide with a brief hypothesis on why the increase/decrease in performance occurred.* ◊

**Problem 6** *Could you outperform the baseline networks? Explain what, in your opinion, contributed to the improved performance.* ◊

**Problem 7** *Experiment with different learning rates. Demonstrate its effects by plotting the loss during training and a short explanation of it.* ◊