HW2 – Report

Gal Barak – 204233688

Shahar Stahi – 305237257

Program structure and run instructions:

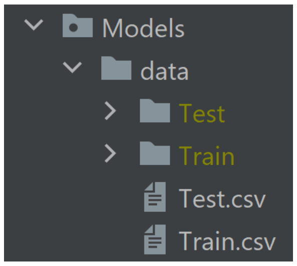
The program contains 2 python packages:

Models:

Contains the models architecture, data and train phases.

To build and train the models run the main function in *Models/main.py*

Important: The program expects to have the data in */data* directory in the following way:



```
Models
  data
    Test
    Train
    Test.csv
    Train.csv
```

We included the .csv files but left the data itself outside the project because it's size. This is not a requirement to run the GUI.

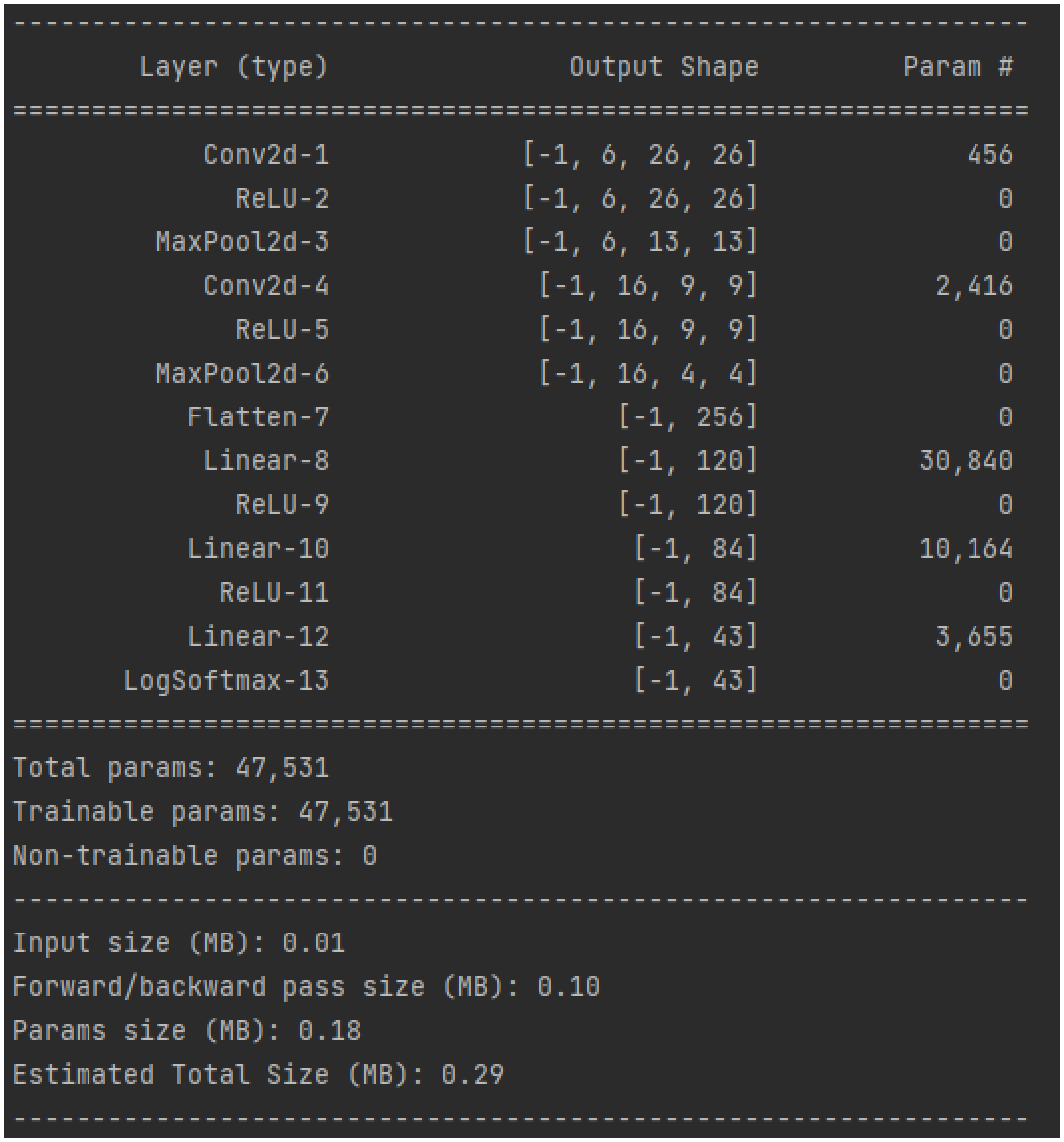
GUI:

Contains the GUI part, on initiating it loads the 3rd model and will produce predictions based on it.

To run the GUI simply run the main function in *main_GUI.py*

Model architecture description

Model 1:



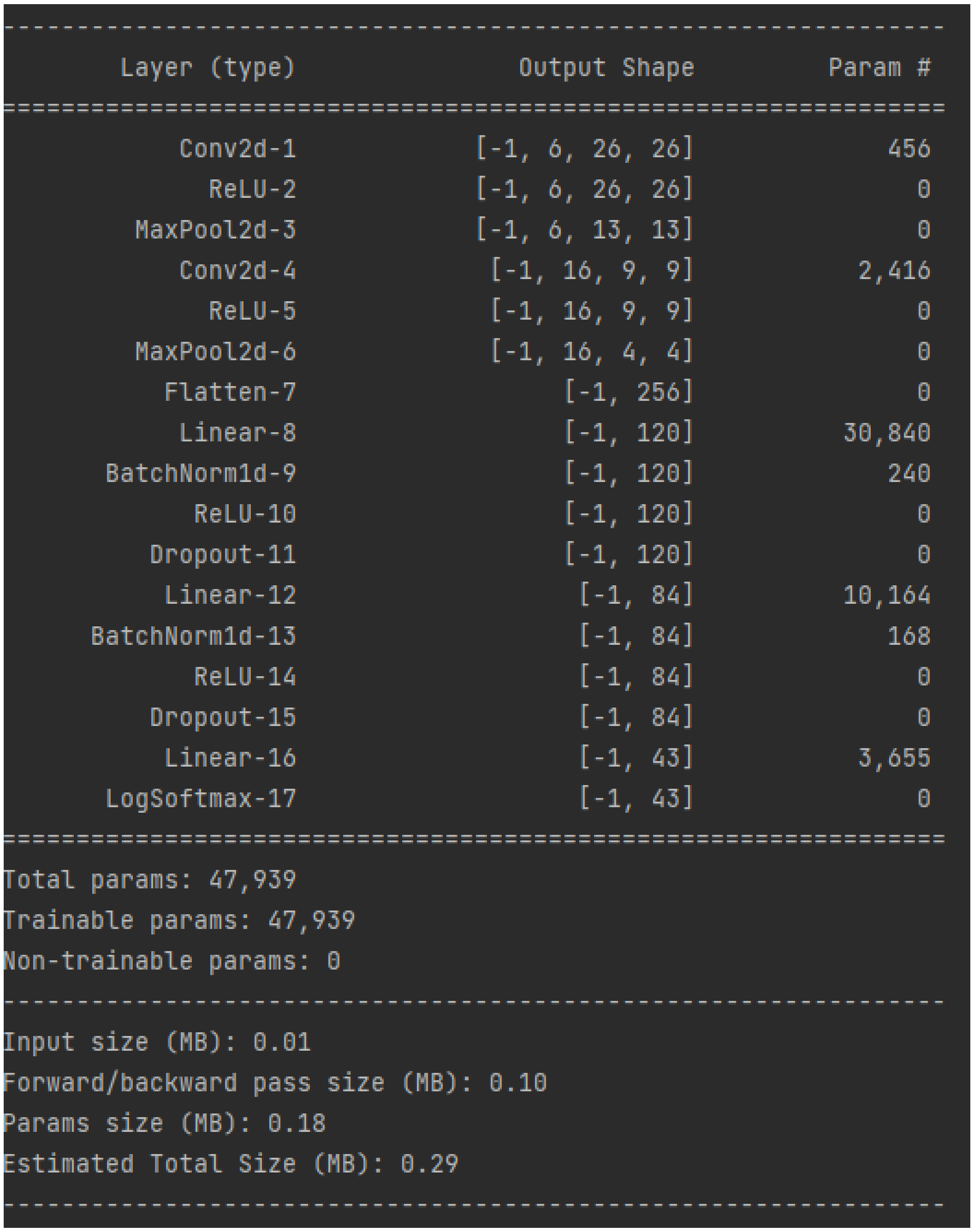
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 26, 26]             456
              ReLU-2            [-1, 6, 26, 26]               0
         MaxPool2d-3            [-1, 6, 13, 13]               0
            Conv2d-4             [-1, 16, 9, 9]           2,416
              ReLU-5             [-1, 16, 9, 9]               0
         MaxPool2d-6             [-1, 16, 4, 4]               0
           Flatten-7                  [-1, 256]               0
            Linear-8                  [-1, 120]          30,840
              ReLU-9                  [-1, 120]               0
           Linear-10                   [-1, 84]          10,164
             ReLU-11                   [-1, 84]               0
           Linear-12                   [-1, 43]           3,655
       LogSoftmax-13                   [-1, 43]               0
================================================================
Total params: 47,531
Trainable params: 47,531
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.10
Params size (MB): 0.18
Estimated Total Size (MB): 0.29
----------------------------------------------------------------
```

The first model is based on LeNet as we studied in the practicals.

The idea was to create a basic network in terms of complexity and number of parameters, in order to create an initial benchmark for the next models.

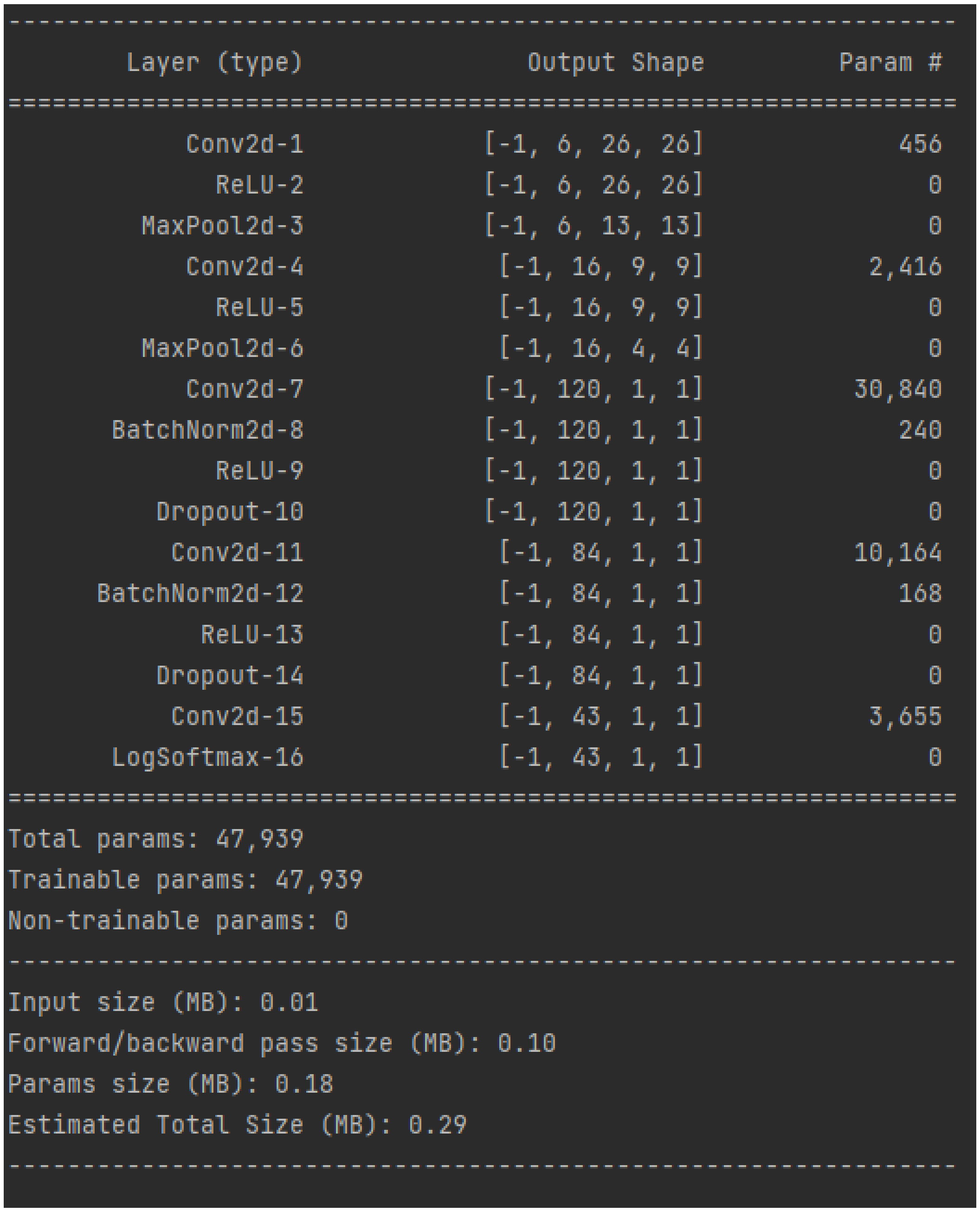
The first model provided a good initial results (as we will explain in more detail later)

Model 2:


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 26, 26]             456
              ReLU-2            [-1, 6, 26, 26]               0
         MaxPool2d-3            [-1, 6, 13, 13]               0
            Conv2d-4             [-1, 16, 9, 9]           2,416
              ReLU-5             [-1, 16, 9, 9]               0
         MaxPool2d-6             [-1, 16, 4, 4]               0
           Flatten-7                  [-1, 256]               0
            Linear-8                  [-1, 120]          30,840
       BatchNorm1d-9                  [-1, 120]             240
             ReLU-10                  [-1, 120]               0
          Dropout-11                  [-1, 120]               0
           Linear-12                   [-1, 84]          10,164
      BatchNorm1d-13                   [-1, 84]             168
             ReLU-14                   [-1, 84]               0
          Dropout-15                   [-1, 84]               0
           Linear-16                   [-1, 43]           3,655
       LogSoftmax-17                   [-1, 43]               0
================================================================
Total params: 47,939
Trainable params: 47,939
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.10
Params size (MB): 0.18
Estimated Total Size (MB): 0.29
----------------------------------------------------------------
```

The second model was based on the first model with the addition of dropout and batch normalization. Similarly to the first model, we used a fully connected classifier.

Model 3:



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 26, 26]             456
              ReLU-2            [-1, 6, 26, 26]               0
         MaxPool2d-3            [-1, 6, 13, 13]               0
            Conv2d-4             [-1, 16, 9, 9]           2,416
              ReLU-5             [-1, 16, 9, 9]               0
         MaxPool2d-6             [-1, 16, 4, 4]               0
            Conv2d-7            [-1, 120, 1, 1]          30,840
       BatchNorm2d-8            [-1, 120, 1, 1]             240
              ReLU-9            [-1, 120, 1, 1]               0
          Dropout-10            [-1, 120, 1, 1]               0
           Conv2d-11             [-1, 84, 1, 1]          10,164
      BatchNorm2d-12             [-1, 84, 1, 1]             168
             ReLU-13             [-1, 84, 1, 1]               0
          Dropout-14             [-1, 84, 1, 1]               0
           Conv2d-15             [-1, 43, 1, 1]           3,655
       LogSoftmax-16             [-1, 43, 1, 1]               0
================================================================
Total params: 47,939
Trainable params: 47,939
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.10
Params size (MB): 0.18
Estimated Total Size (MB): 0.29
----------------------------------------------------------------
```

The third model is a fully convolutional network.

The first part (feature extractor) is similar to the 2nd model but the classifier has convolution layers with different kernel sizes and ends with a 1x1 convolution.

training procedure

The process has a limit of 120 epochs and usually performs early stopping around 80-90 epochs.

In each epoch we use iterations (batch) size of 128 samples.

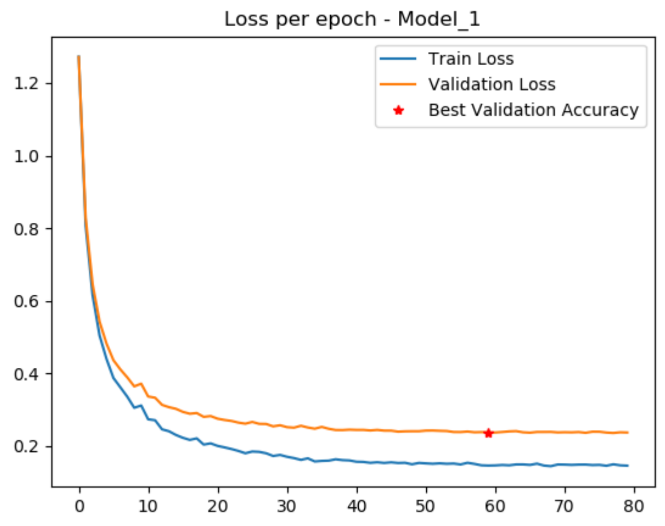
The training process includes a feedforward flow, loss calculation and backpropagation.

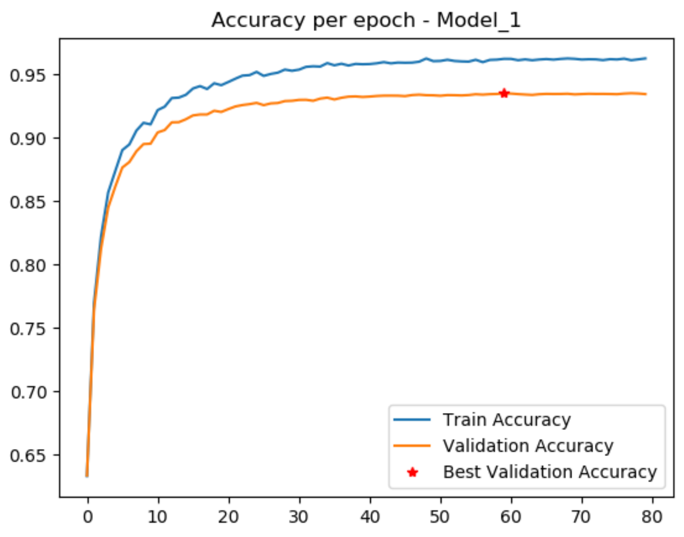
As a preliminary step was a creation of different augmentations (both random and deterministic ones) in order to increase our data.

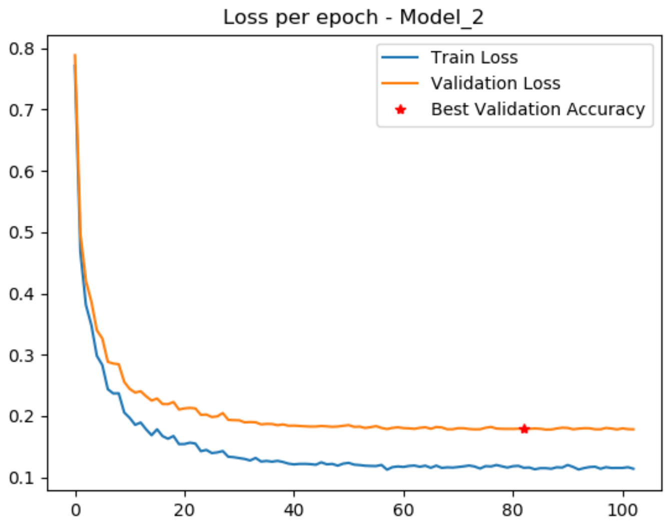
Accuracy and loss as a function of time (epochs)

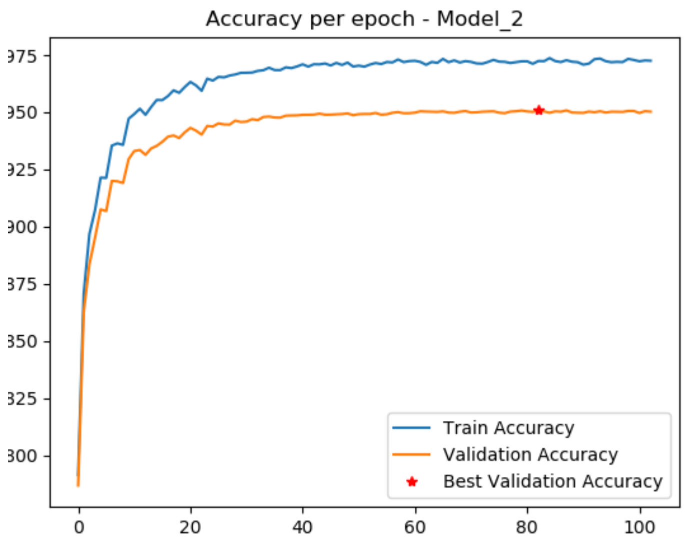
Below are the final results we reached in our training process.

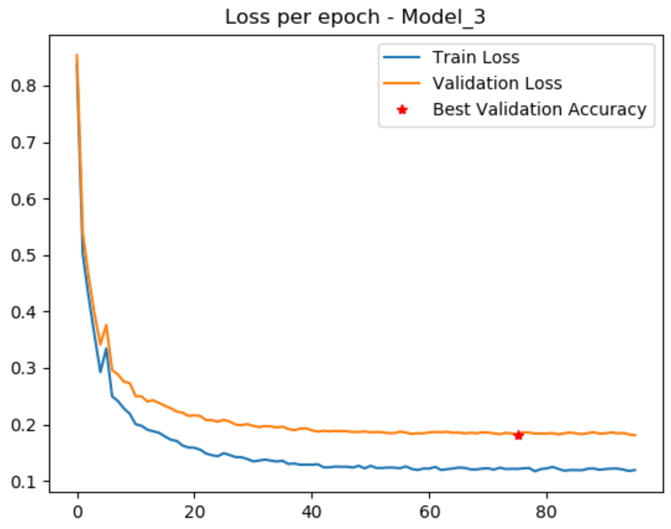
It is important to note that these results were achieved by using random augmentations and hence weren't always stable (+- 1%) but the picture below gives a good representation of the average case

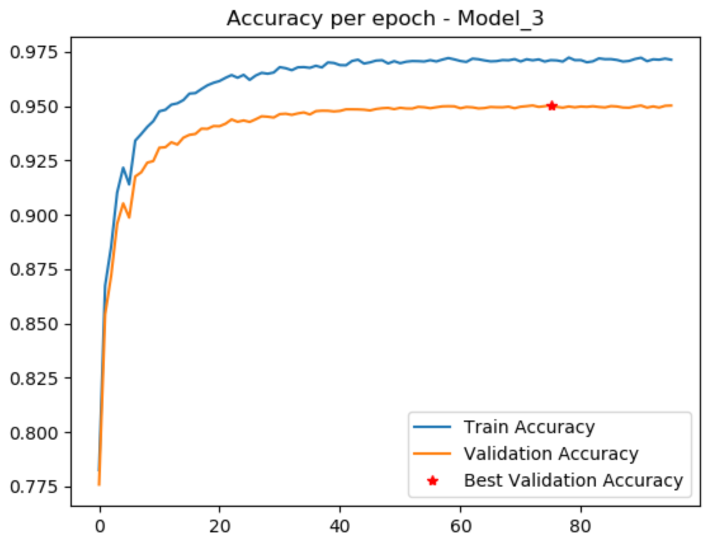




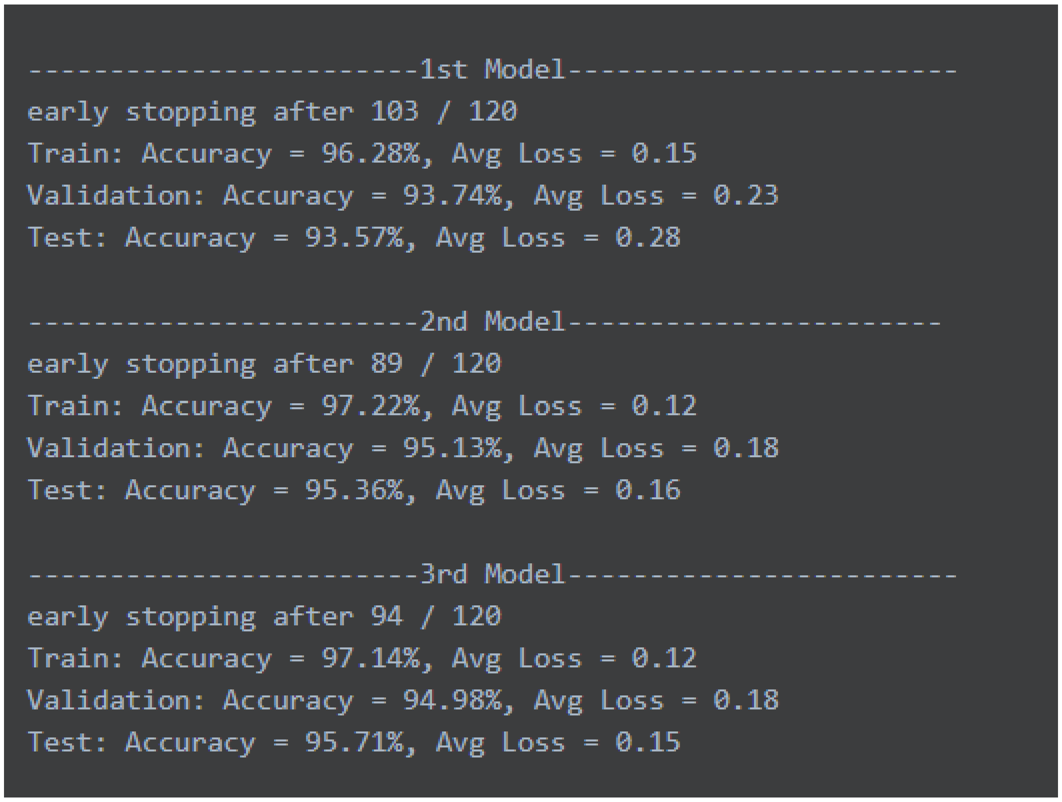

Loss per epoch - Model_2
Train Loss
Validation Loss
Best Validation Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100


Accuracy per epoch - Model_2
975
950
925
900
875
850
825
800
Train Accuracy
Validation Accuracy
Best Validation Accuracy
0
20
40
60
80
100


Loss per epoch - Model_3
Train Loss
Validation Loss
Best Validation Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80


Accuracy per epoch - Model_3
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.775
Train Accuracy
Validation Accuracy
Best Validation Accuracy
0
20
40
60
80

Models test accuracies



```
-----------------------1st Model-----------------------
early stopping after 103 / 120
Train: Accuracy = 96.28%, Avg Loss = 0.15
Validation: Accuracy = 93.74%, Avg Loss = 0.23
Test: Accuracy = 93.57%, Avg Loss = 0.28

-----------------------2nd Model----------------------
early stopping after 89 / 120
Train: Accuracy = 97.22%, Avg Loss = 0.12
Validation: Accuracy = 95.13%, Avg Loss = 0.18
Test: Accuracy = 95.36%, Avg Loss = 0.16

-----------------------3rd Model-----------------------
early stopping after 94 / 120
Train: Accuracy = 97.14%, Avg Loss = 0.12
Validation: Accuracy = 94.98%, Avg Loss = 0.18
Test: Accuracy = 95.71%, Avg Loss = 0.15
```

As we can see, the first model needed more epochs in order to maximize its abilities, as a contrary to the other two which got their best results faster by using advanced methods of regularizations.

What did we try?

The development process included a number of steps.

First, as we explained in the beginning, we based our model on LeNet.

Then we started a process of tuning our parameters.

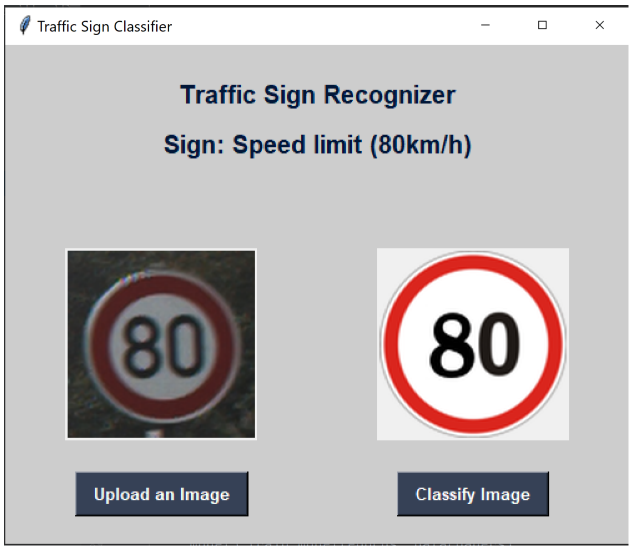
We learned the weight decay and learning rate in our optimizer, dropout probability in the dropout mechanism and momentum for the batch normalization process.

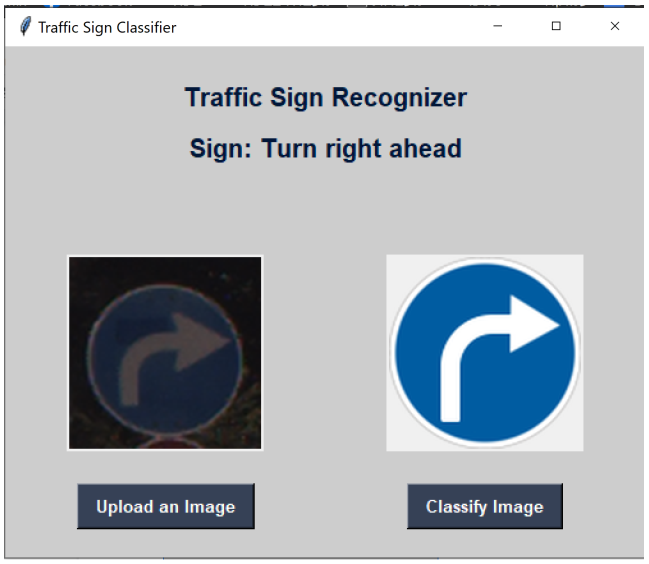
After learning and tuning our parameters we try 3 batch sizes. The chosen size was 128 which gave the best results in terms of running time after adding our augmentations.

In the development of the 3rd model we tried many additions / removals / changes in the classifier layers. The layers which produced the best results can be seen in detail in the beginning of the report.

In conclusion, our main consideration in the process was to keep a good ratio between the model's simplicity and the network's results on the validation set. Doing so helps us ensure we avoid overfitting and helps the network to take on a wider and more complex test set without the need of special adjustments.

<u>GUI screenshots</u>


Traffic Sign Classifier
Traffic Sign Recognizer
Sign: Speed limit (80km/h)
80
80
Upload an Image
Classify Image


Traffic Sign Classifier
Traffic Sign Recognizer
Sign: Turn right ahead
Upload an Image
Classify Image