Final Project – Report

Gal Barak – 204233688

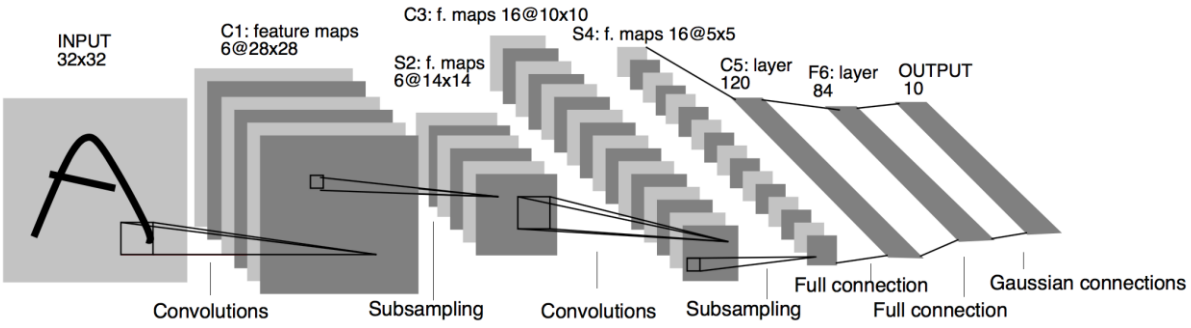
Shahar Stahi – 305237257

Main Goal

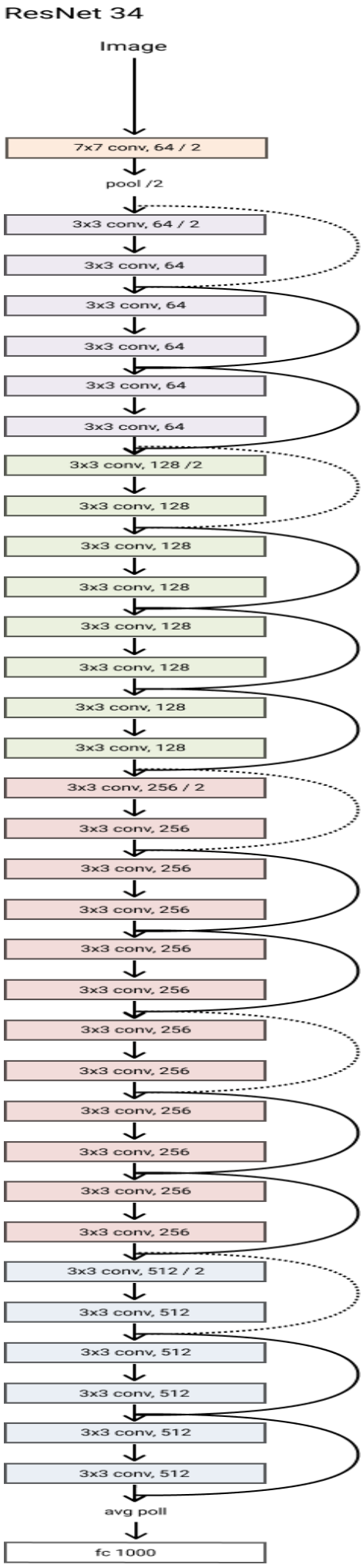
To compare different classifying networks on the same input in order to find the network that provides the best results by taking into account the net's complexity (number of parameters) and training time.

The networks we used:

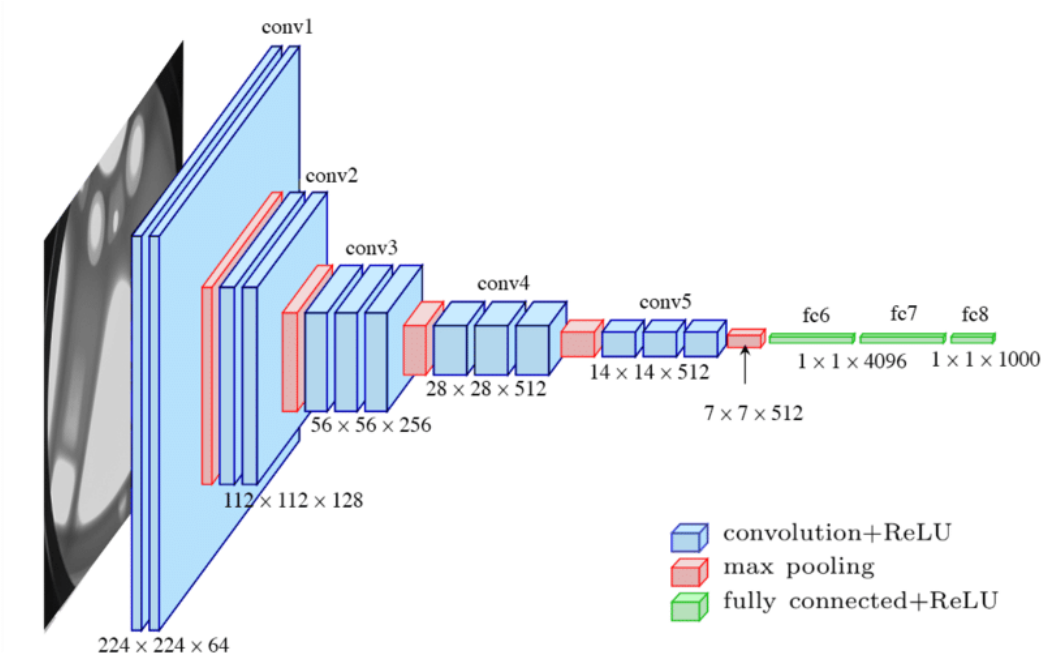
LeNet5:



ResNet34:


ResNet 34
Image
7x7 conv, 64 / 2
pool /2
3x3 conv, 64 / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128 /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256 / 2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512 / 2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg poll
fc 1000

VGG16



The networks were trained on the same dataset – 'German Traffic Sign Recognition Benchmark'

The input size for each network was 32X32. In addition, we created different augmentations in order to increase the dataset size and its variance (i.e brightness, cropping, grayscale conversion and more)

The networks implementation was done in their standard way, no major differences were made in the networks themselves in order to improve their basic results.

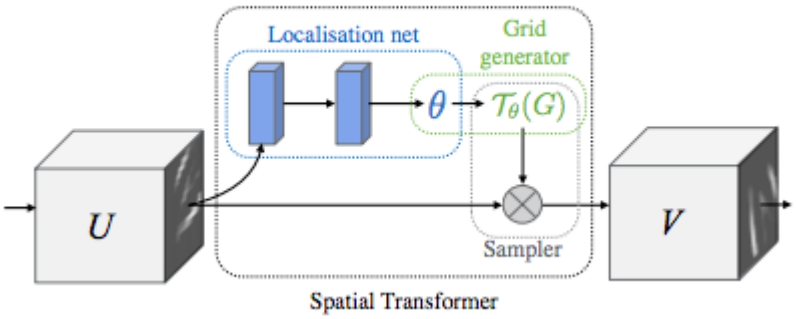
Then, we used Spatial Transformations to improve the net's performances.

Spatial transformer networks:

Spatial transformer networks are a generalization of differentiable attention to any spatial transformation. Spatial transformer networks (STN for short) allow a neural network to learn how to perform spatial transformations on the input image in order to enhance the geometric invariance of the model. For example, it can crop a region of interest, scale and correct the orientation of an image. It can be a useful mechanism because CNNs are not invariant to rotation and scale and more general affine transformations.

Spatial transformer networks boils down to three main components:

- The localization network is a regular CNN which regresses the transformation parameters. The transformation is never learned explicitly from this dataset, instead the network learns automatically the spatial transformations that enhances the global accuracy.
- The grid generator generates a grid of coordinates in the input image corresponding to each pixel from the output image.
- The sampler uses the parameters of the transformation and applies it to the input image.



Spatial Transformer

```
  (localization): Sequential(
    (0): Conv2d(3, 8, kernel_size=(7, 7), stride=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): ReLU(inplace=True)
    (3): Conv2d(8, 10, kernel_size=(5, 5), stride=(1, 1))
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): ReLU(inplace=True)
  )
  (fc_loc): Sequential(
    (0): Linear(in_features=160, out_features=32, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=32, out_features=6, bias=True)
  )

----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 8, 26, 26]           1,184
         MaxPool2d-2           [-1, 8, 13, 13]               0
              ReLU-3           [-1, 8, 13, 13]               0
            Conv2d-4           [-1, 10, 9, 9]            2,010
         MaxPool2d-5           [-1, 10, 4, 4]               0
              ReLU-6           [-1, 10, 4, 4]               0
            Linear-7                 [-1, 32]           5,152
              ReLU-8                 [-1, 32]               0
            Linear-9                  [-1, 6]             198
================================================================
```

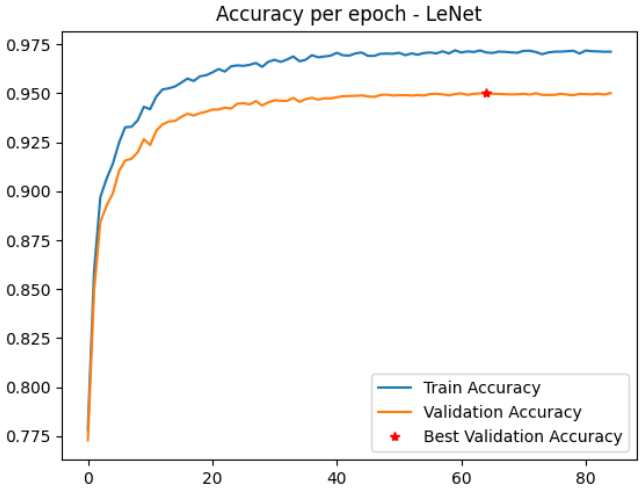
Training Results:

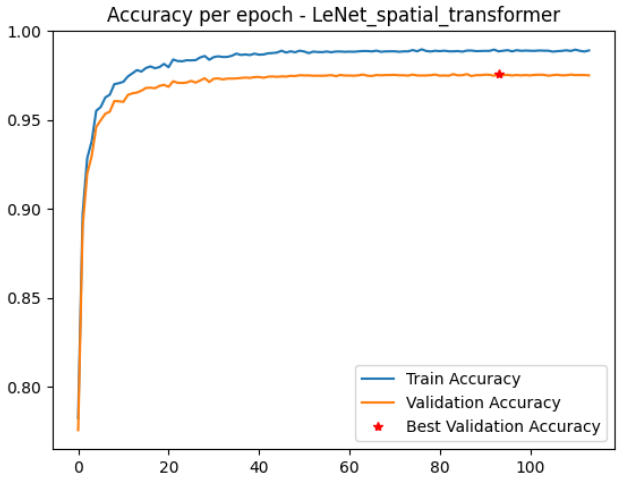
Training Results

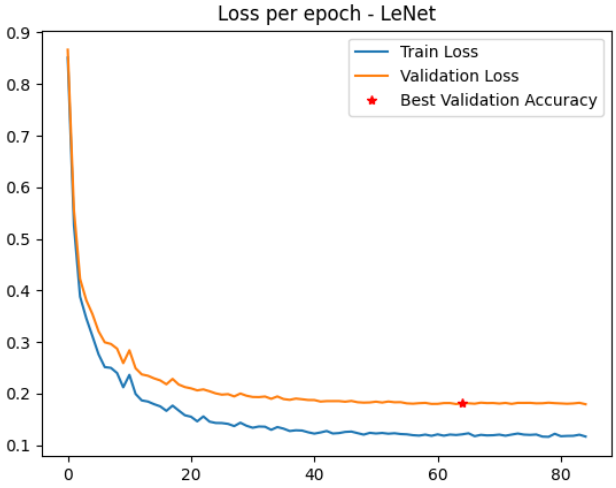
```
--------------------------- LeNet 5 ---------------------------
Train: Accuracy = 97.54%, Avg Loss = 0.10
Validation: Accuracy = 94.99%, Avg Loss = 0.18
Test: Accuracy = 95.69%, Avg Loss = 0.16

        Time taken: 01:00:40.54

--------------------------- VGG 16 ----------------------------
Train: Accuracy = 99.72%, Avg Loss = 0.01
Validation: Accuracy = 98.45%, Avg Loss = 0.07
Test: Accuracy = 97.76%, Avg Loss = 0.13

        Time taken: 02:52:46.96

------------------------- ResNet 34 ----------------------------
Train: Accuracy = 99.81%, Avg Loss = 0.01
Validation: Accuracy = 98.90%, Avg Loss = 0.05
Test: Accuracy = 98.83%, Avg Loss = 0.05

        Time taken: 01:36:23.83
```

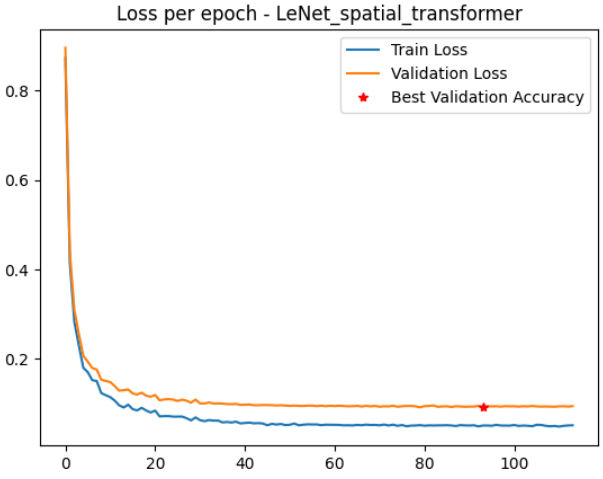
Training Results with Spatial Transformer

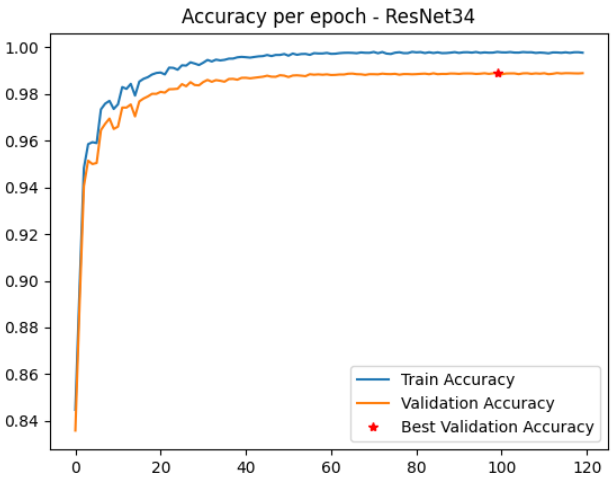
```
--------------------------- LeNet 5 ---------------------------
Train: Accuracy = 98.86%, Avg Loss = 0.05
Validation: Accuracy = 97.60%, Avg Loss = 0.09
Test: Accuracy = 98.15%, Avg Loss = 0.08

        Time taken: 01:26:54.03

--------------------------- VGG 16 ----------------------------
Train: Accuracy = 99.38%, Avg Loss = 0.02
Validation: Accuracy = 97.25%, Avg Loss = 0.11
Test: Accuracy = 96.07%, Avg Loss = 0.20

        Time taken: 02:21:11.51

------------------------- ResNet 34 ----------------------------
Train: Accuracy = 99.72%, Avg Loss = 0.01
Validation: Accuracy = 98.88%, Avg Loss = 0.05
Test: Accuracy = 98.82%, Avg Loss = 0.05

        Time taken: 01:50:42.37
```

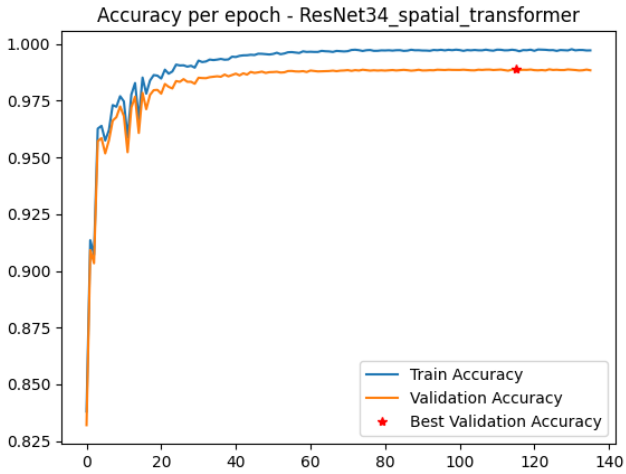

Accuracy per epoch - LeNet
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0
20
40
60
80
Train Accuracy
Validation Accuracy
Best Validation Accuracy

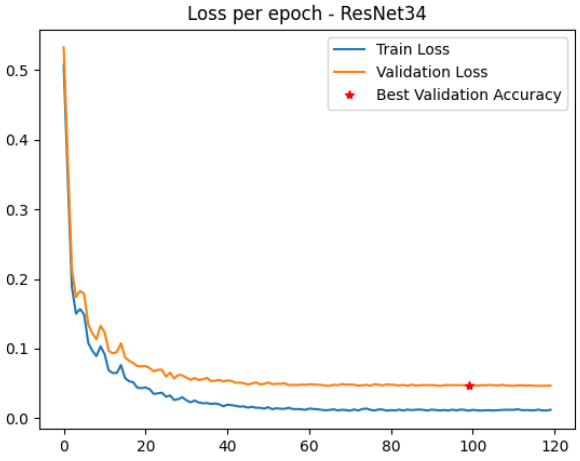

Accuracy per epoch - LeNet_spatial_transformer
1.00
0.95
0.90
0.85
0.80
0
20
40
60
80
100
Train Accuracy
Validation Accuracy
Best Validation Accuracy

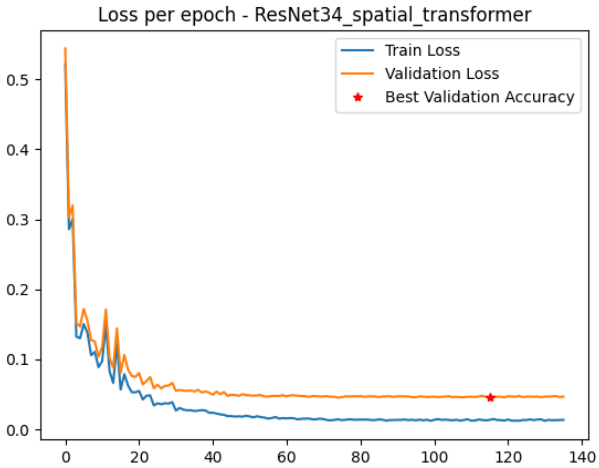

Loss per epoch - LeNet
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
Train Loss
Validation Loss
Best Validation Accuracy

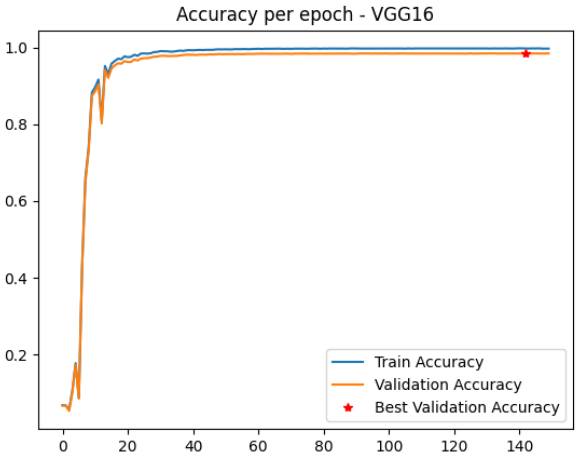

Loss per epoch - LeNet_spatial_transformer
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Train Loss
Validation Loss
Best Validation Accuracy

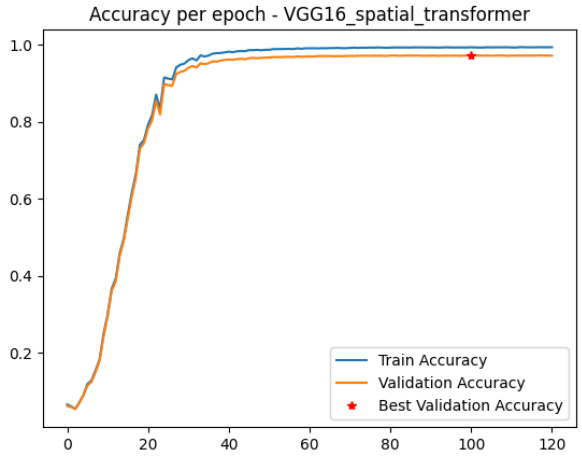

Accuracy per epoch - ResNet34
Train Accuracy
Validation Accuracy
Best Validation Accuracy

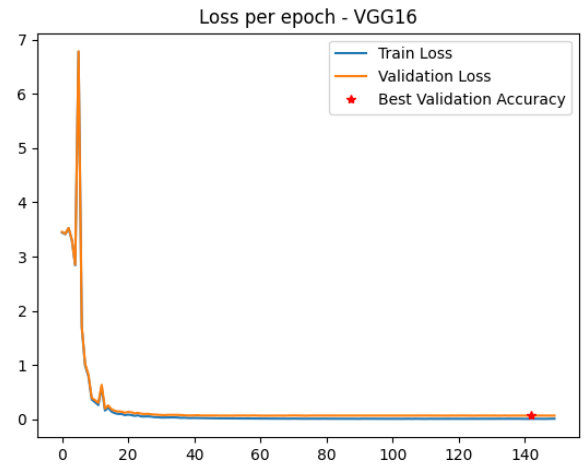

Accuracy per epoch - ResNet34_spatial_transformer
Train Accuracy
Validation Accuracy
Best Validation Accuracy

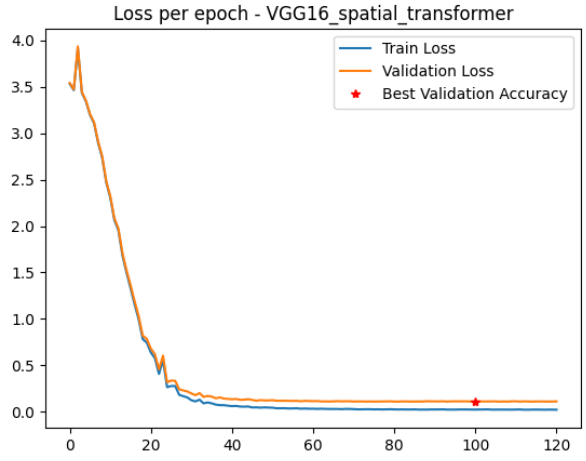

Loss per epoch - ResNet34
Train Loss
Validation Loss
Best Validation Accuracy


Loss per epoch - ResNet34_spatial_transformer
Train Loss
Validation Loss
Best Validation Accuracy


Accuracy per epoch - VGG16
Train Accuracy
Validation Accuracy
Best Validation Accuracy


Accuracy per epoch - VGG16_spatial_transformer
Train Accuracy
Validation Accuracy
Best Validation Accuracy


Loss per epoch - VGG16
Train Loss
Validation Loss
Best Validation Accuracy


Loss per epoch - VGG16_spatial_transformer
Train Loss
Validation Loss
Best Validation Accuracy

Train and test summery and conclusions:

As can be seen, we used 3 different classifying networks.

LeNet – the simplest one with 5 convolution layers and around 65k parameters produced results of 95.69% (for the basic network) and 98.15% (with STN addition)

ResNet – a more advanced network with 34 convolution layers and around 10m parameters produced results of 98.83 for the regular network and 98.82% with STN addition. That is to say, in the ResNet case, adding STN didn't improve the test results and was actually wasteful – It added more parameters and took a longer time to train (by 20 mins on average). ResNet train times were between 1:00 to 1:20 hours.

VGG – The most complex network which includes 16 convolution layers and more than 130m parameters. Train times were between 2:20 to 2:50 hours.

VGG results without STN were 97.76% and 97.25% with it. In a similar way to ResNet using STN didn't improve the test results but it did benefit it in a different way. Using VGG with STN helped the training process to reach the 'early stopping' point faster and hence reduced the train time by 30 mins on average. Unlike LeNet and ResNet which took longer time to train with STN.

Final conclusions:

All the network we tested produced results of over 96% on the test set. LeNet (with STN) and ResNet (with and without STN) even passed the 98% mark.

VGG which is the most complex network in terms of parameters and train time gave inferior results and took more time. Our assumption is that VGG is designed to a more complex tasks and hence the high number of parameters led to inferior results in a relatively simple task as classifying 32X32 traffic signs.

When comparing LeNet and ResNet, ResNet did produced better results with and without STN but it's running time took about half an hour more (1.5X LeNet's time)

We also added some graphs to compare between the models' size, train time and number of parameters.