Assignment 2

Kaggle Competition GTSRB

Aishwarya Budhkar

1) Summary

I tried to experiment with the following models

- Used 3- layer Convolutional Network as base test
- Used ResNet 18. Modified the same file for ResNet with STN
- Used VGG. Tried to implement VGG. Accuracy achieved is 97%
- Used Spatial Transformer Network modules with modified layers
- Used Spatial Transformer Network modules with ResNet18

2) Introduction

Assignment is to develop and train models which will be used to classify signs in Germen Traffic Signs Recognition Benchmark [1] dataset. Starter code and test and train images are provided.

3) Preprocessing

- Rescale- Rescaled the images to 48x48 and 64x64. When rescaled the images to 48X48, the results were better than 32x32.
- Transform- Tried to use rotate Transform for ResNet18 but didn't get good results.

4) Models Implemented

Model 1:

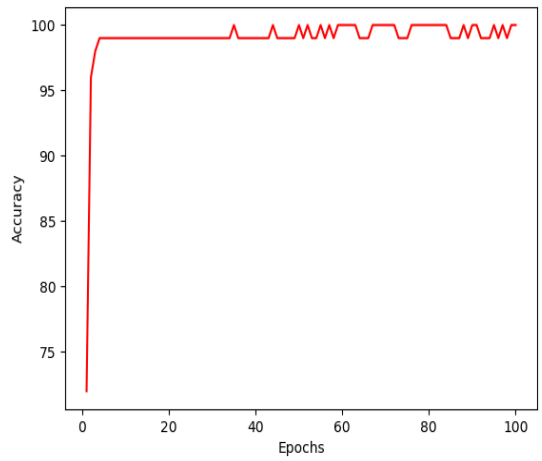
Tried the basic 2-layer model given on course website. Trained on data to achieve accuracy 92%. Then trained a simple 3-layer convolutional network with 2 fully connected layers and high number of channels (100, 150 and 250). Achieved a test accuracy of around 97.7% with this model.

Optimizer- Stochastic Gradient Descent
Learning rate – 0.1

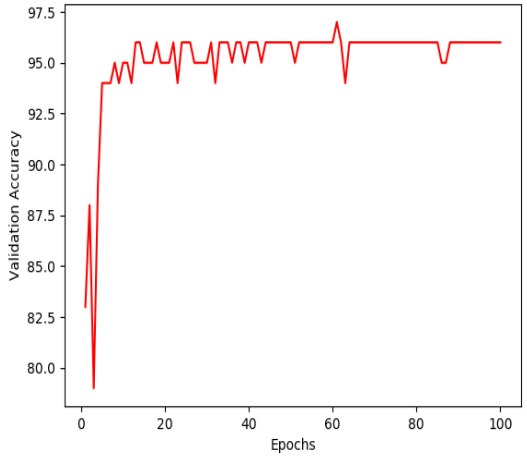
Improvement:
Added batch normalization layers to this CNN. Changed learning rate to 0.01. Increased Image size to 48. Got best accuracy of 98.828%

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 100, 42, 42]          14,800
       BatchNorm2d-2          [-1, 100, 42, 42]             200
            Conv2d-3          [-1, 150, 18, 18]         240,150
       BatchNorm2d-4          [-1, 150, 18, 18]             300
            Conv2d-5            [-1, 250, 6, 6]         600,250
       BatchNorm2d-6            [-1, 250, 6, 6]             500
            Linear-7                  [-1, 300]         675,300
            Linear-8                   [-1, 43]          12,943
================================================================
Total params: 1,544,443
Trainable params: 1,544,443
Non-trainable params: 0
----------------------------------------------------------------
```

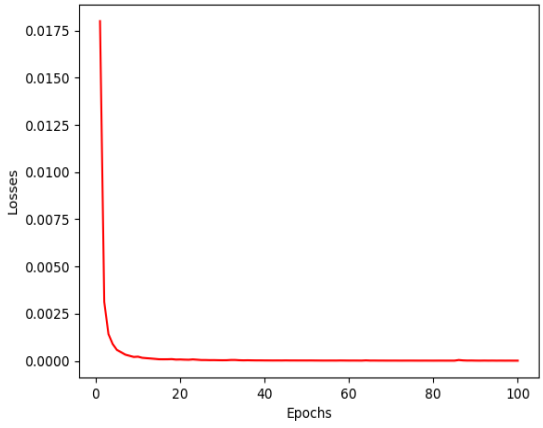
Training accuracy:



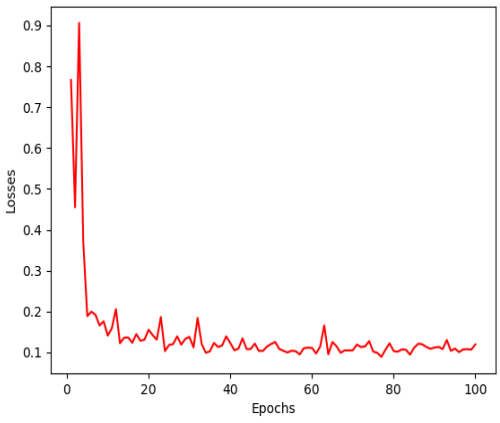
Validation accuracy:



Training loss:



Validation loss:



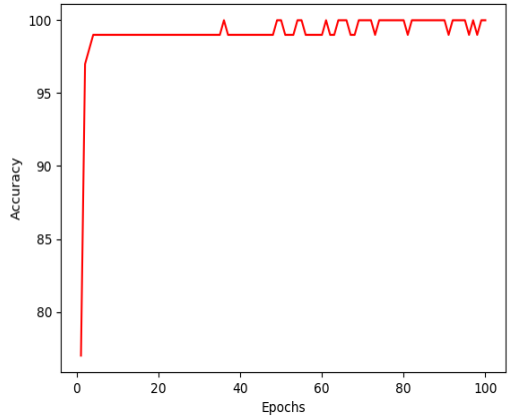
Model 2:

Increased number of neurons in output layer. Trained a simple 3-layer convolutional network with 2 fully connected layers and high number of channels (150, 200 and 300). Increased Image size to 64. I got an accuracy of 97.672%.

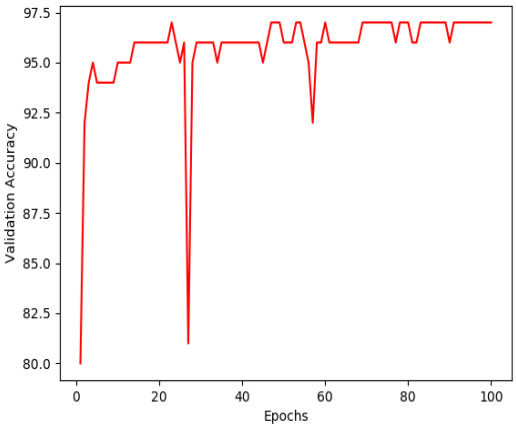
Optimizer - SGD
Learning rate- 0.01

```
----------------------------------------------------------------
Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 100, 58, 58]          14,800
       BatchNorm2d-2         [-1, 100, 58, 58]             200
            Conv2d-3         [-1, 150, 26, 26]         240,150
       BatchNorm2d-4         [-1, 150, 26, 26]             300
            Conv2d-5         [-1, 250, 10, 10]         600,250
       BatchNorm2d-6         [-1, 250, 10, 10]             500
            Linear-7                [-1, 350]       2,187,850
            Linear-8                 [-1, 43]          15,093
================================================================
Total params: 3,059,143
Trainable params: 3,059,143
Non-trainable params: 0
```

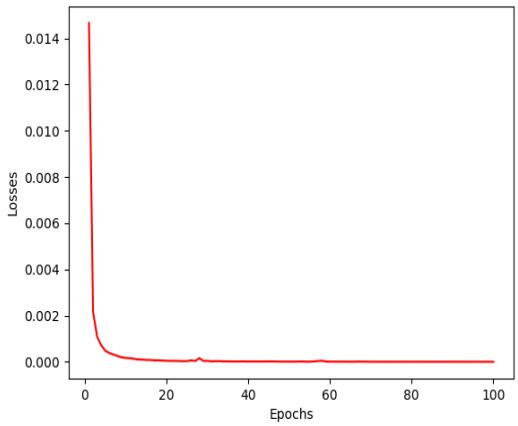
Training accuracy:



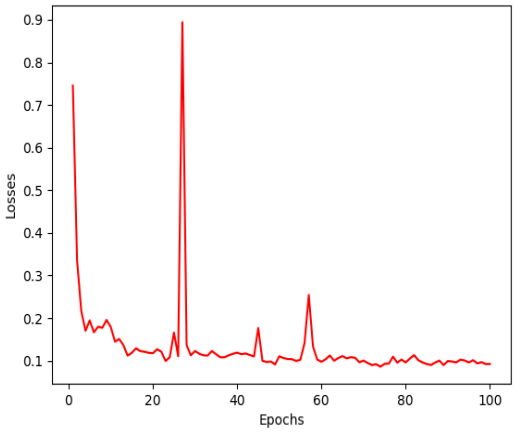
Validation accuracy:



Training loss:



Validation loss:



Increase in the image size didn't help to increase the accuracy. Training was slower. Tried to change the momentum to 0.7. But the accuracy didn't change much though the convergence was better.

Model 3:

Spatial Transformer Network – Currently, the state of the art results are held by a model [2] which uses Spatial Transformer Networks [3]. Both local and global features through spatial normalization (both locally and globally). STNs are used to avoid data augmentation and take care of color and spatial variance of image. Performs explicit geometric transformation for images.

Added batch normalization and drop out to improve the accuracy by normalizing the inputs.

Learning rate 0.001

Learning rate decay: 0.00001

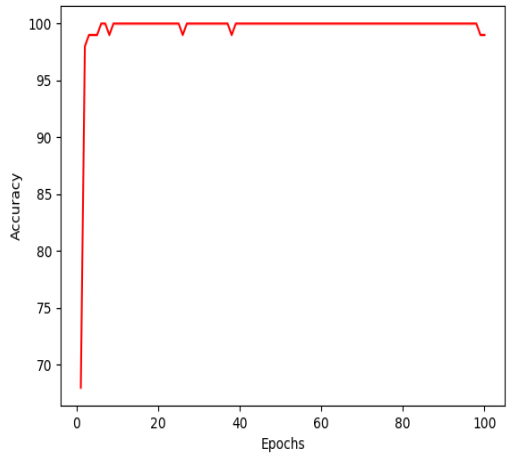
Optimizer- Adam

Accuracy: 98.986%

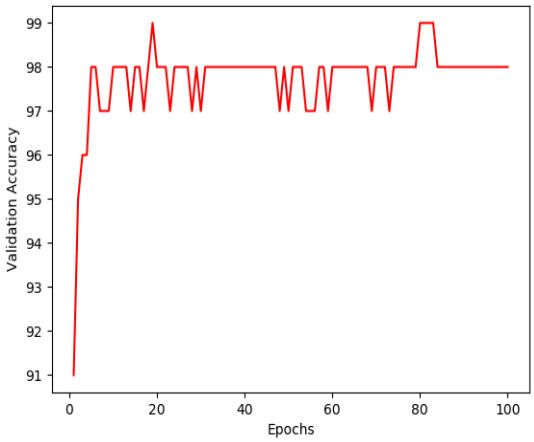
```
----------------------------------------------------------------
Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 10, 52, 52]              40
         Dropout2d-2          [-1, 10, 52, 52]               0
       BatchNorm2d-3          [-1, 10, 52, 52]              20
            Conv2d-4           [-1, 3, 56, 56]              33
         Dropout2d-5           [-1, 3, 56, 56]               0
       BatchNorm2d-6           [-1, 3, 56, 56]               6
            Conv2d-7          [-1, 16, 52, 52]           2,368
            Conv2d-8          [-1, 32, 24, 24]          12,832
            Conv2d-9          [-1, 64, 12, 12]          18,496
           Linear-10                [-1, 128]         295,040
           Linear-11                 [-1, 64]           8,256
           Linear-12                  [-1, 6]             390
           Conv2d-13          [-1, 16, 48, 48]           1,216
```

```
        Dropout2d-14          [-1, 16, 48, 48]               0
      BatchNorm2d-15          [-1, 16, 48, 48]              32
           Conv2d-16          [-1, 32, 48, 48]          12,832
        Dropout2d-17          [-1, 32, 48, 48]               0
      BatchNorm2d-18          [-1, 32, 48, 48]              64
           Conv2d-19          [-1, 64, 48, 48]          51,264
        Dropout2d-20          [-1, 64, 48, 48]               0
      BatchNorm2d-21          [-1, 64, 48, 48]             128
           Conv2d-22          [-1, 96, 48, 48]         153,696
        Dropout2d-23          [-1, 96, 48, 48]               0
      BatchNorm2d-24          [-1, 96, 48, 48]             192
           Conv2d-25         [-1, 128, 48, 48]         307,328
        Dropout2d-26         [-1, 128, 48, 48]               0
      BatchNorm2d-27         [-1, 128, 48, 48]             256
           Conv2d-28         [-1, 192, 48, 48]         614,592
        Dropout2d-29         [-1, 192, 48, 48]               0
      BatchNorm2d-30         [-1, 192, 48, 48]             384
           Conv2d-31         [-1, 256, 24, 24]       1,229,056
        Dropout2d-32         [-1, 256, 24, 24]               0
      BatchNorm2d-33         [-1, 256, 24, 24]             512
           Conv2d-34         [-1, 128, 12, 12]         819,328
        Dropout2d-35         [-1, 128, 12, 12]               0
      BatchNorm2d-36         [-1, 128, 12, 12]             256
           Conv2d-37          [-1, 64, 12, 12]         204,864
        Dropout2d-38          [-1, 64, 12, 12]               0
      BatchNorm2d-39          [-1, 64, 12, 12]             128
           Linear-40                 [-1, 43]           2,795
        Dropout2d-41                 [-1, 43]               0
================================================================
Total params: 3,736,404
Trainable params: 3,736,404
Non-trainable params: 0
```

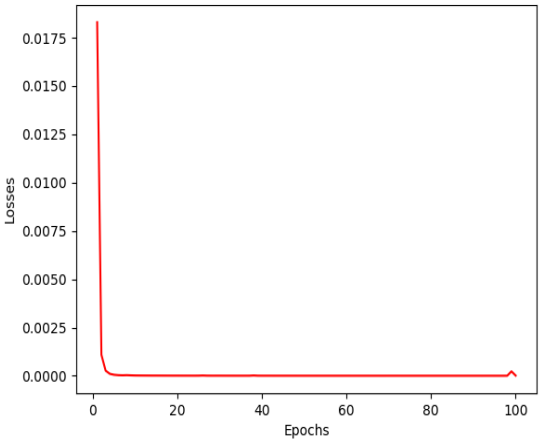
Training accuracy:



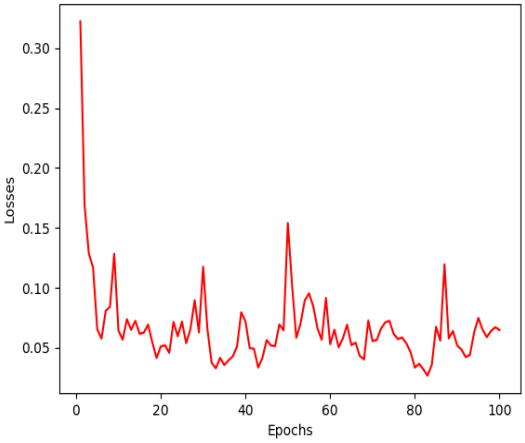
Testing accuracy:



Training loss:



Testing loss:



Model 4:

Tried to implement ResNet 18.

Learning rate – 0.0001

Optimizer-Adam

Accuracy achieved was 98.957%.

So, tried to use Spatial Transform Network with Resnet 18:

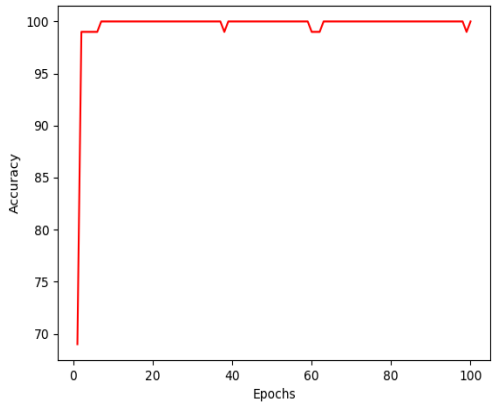
Image size: 48

Got Test accuracy: 98.337%

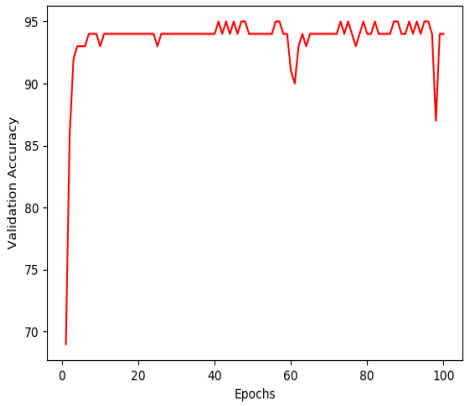
```
----------------------------------------------------------------
Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 10, 52, 52]              40
         Dropout2d-2          [-1, 10, 52, 52]               0
       BatchNorm2d-3          [-1, 10, 52, 52]              20
            Conv2d-4           [-1, 3, 56, 56]              33
         Dropout2d-5           [-1, 3, 56, 56]               0
       BatchNorm2d-6           [-1, 3, 56, 56]               6
            Conv2d-7          [-1, 16, 52, 52]           2,368
            Conv2d-8          [-1, 32, 24, 24]          12,832
            Conv2d-9          [-1, 64, 12, 12]          18,496
           Linear-10                [-1, 128]         295,040
           Linear-11                 [-1, 64]           8,256
           Linear-12                  [-1, 6]             390
           Conv2d-13          [-1, 64, 24, 24]           9,408
      BatchNorm2d-14          [-1, 64, 24, 24]             128
```

```
         MaxPool2d-15           [-1, 64, 12, 12]               0
             Conv2d-16           [-1, 64, 12, 12]          36,864
        BatchNorm2d-17           [-1, 64, 12, 12]             128
             Conv2d-18           [-1, 64, 12, 12]          36,864
        BatchNorm2d-19           [-1, 64, 12, 12]             128
         BasicBlock-20           [-1, 64, 12, 12]               0
             Conv2d-21           [-1, 64, 12, 12]          36,864
        BatchNorm2d-22           [-1, 64, 12, 12]             128
             Conv2d-23           [-1, 64, 12, 12]          36,864
        BatchNorm2d-24           [-1, 64, 12, 12]             128
         BasicBlock-25           [-1, 64, 12, 12]               0
             Conv2d-26            [-1, 128, 6, 6]          73,728
        BatchNorm2d-27            [-1, 128, 6, 6]             256
             Conv2d-28            [-1, 128, 6, 6]         147,456
        BatchNorm2d-29            [-1, 128, 6, 6]             256
             Conv2d-30            [-1, 128, 6, 6]           8,192
        BatchNorm2d-31            [-1, 128, 6, 6]             256
         BasicBlock-32            [-1, 128, 6, 6]               0
             Conv2d-33            [-1, 128, 6, 6]         147,456
        BatchNorm2d-34            [-1, 128, 6, 6]             256
             Conv2d-35            [-1, 128, 6, 6]         147,456
        BatchNorm2d-36            [-1, 128, 6, 6]             256
         BasicBlock-37            [-1, 128, 6, 6]               0
             Conv2d-38            [-1, 256, 3, 3]         294,912
        BatchNorm2d-39            [-1, 256, 3, 3]             512
             Conv2d-40            [-1, 256, 3, 3]         589,824
        BatchNorm2d-41            [-1, 256, 3, 3]             512
             Conv2d-42            [-1, 256, 3, 3]          32,768
        BatchNorm2d-43            [-1, 256, 3, 3]             512
         BasicBlock-44            [-1, 256, 3, 3]               0
             Conv2d-45            [-1, 256, 3, 3]         589,824
        BatchNorm2d-46            [-1, 256, 3, 3]             512
             Conv2d-47            [-1, 256, 3, 3]         589,824
        BatchNorm2d-48            [-1, 256, 3, 3]             512
         BasicBlock-49            [-1, 256, 3, 3]               0
             Conv2d-50            [-1, 512, 2, 2]       1,179,648
        BatchNorm2d-51            [-1, 512, 2, 2]           1,024
             Conv2d-52            [-1, 512, 2, 2]       2,359,296
        BatchNorm2d-53            [-1, 512, 2, 2]           1,024
             Conv2d-54            [-1, 512, 2, 2]         131,072
        BatchNorm2d-55            [-1, 512, 2, 2]           1,024
         BasicBlock-56            [-1, 512, 2, 2]               0
             Conv2d-57            [-1, 512, 2, 2]       2,359,296
        BatchNorm2d-58            [-1, 512, 2, 2]           1,024
             Conv2d-59            [-1, 512, 2, 2]       2,359,296
        BatchNorm2d-60            [-1, 512, 2, 2]           1,024
         BasicBlock-61            [-1, 512, 2, 2]               0
          AvgPool2d-62            [-1, 512, 1, 1]               0
             Linear-63                  [-1, 43]          22,059
================================================================
Total params: 11,536,052
Trainable params: 11,536,052
Non-trainable params: 0
----------------------------------------------------------------
```

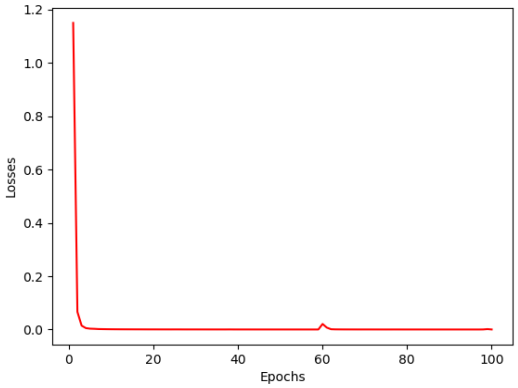
Training accuracy:



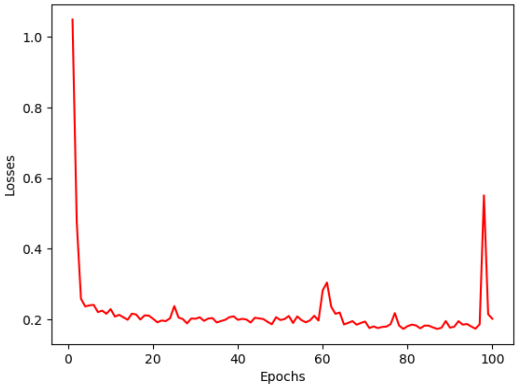
Validation accuracy:



Training loss:



Validation loss:



The model didn't perform as well on test data.

Tried to decrease the learning rate to 5e-5. But the accuracy didn't improve.

5) Results

Model	Image size	Epochs	Learning rate	Optimizer	Accuracy
2-layer CNN	32x32	20	0.1	SGD	92
3-layer CNN	32x32	20	0.01	SGD	94.568
3-layer CNN with Batch Normalization	48x48	100	0.01	SGD	98.828
3-layer CNN with Batch Normalization	48x48	150	0.01	SGD	98.828
3-layer CNN with Batch Normalization	64x64	100	0.01	SGD	97.672

CNN with Spatial Transformer Network	48x48	100	0.01	SGD	98.749
CNN with Spatial Transformer Network	48x48	100	0.001	SGD	98.875
CNN with Spatial Transformer Network	48x48	100	0.001	Adam	98.986
CNN with Spatial Transformer Network	48x48	150	0.00005	Adam	98.337
ResNet with Spatial Transform Network	48x48	100	0.001	Adam	98.606
ResNet with Spatial Transform Network	48x48	150	0.0001	Adam	98.083
ResNet with Spatial Transform Network	48x48	150	0.00005	Adam	98.004
ResNet	48x48	100	0.01	SGD	97.957

6) Conclusion

The images in dataset have severe flaws such as blur, shadow, obstructions, bad lightening. There are distortions like rotation, translation, etc. These are difficult for human to recognize. Spatial Transformer networks are good for classification tasks with lot of distortion in data. As, STNs take care of color and spatial variance and avoid augmentation, with learned color and spatial variance the model can adapt to image's characteristics.

Improvement:

More experimentation with hyper parameters can lead to better results. Also, multiple STNs [4] can be added to get better accuracy. As the current model is trained on very limited dataset, it is unable to recognize other signs which it didn't encounter. With more data, the accuracy might improve.

7) References

[1] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In Neural Networks (IJCNN), The 2011 International Joint Conference on, pages 1453–1460. IEEE, 2011.

[2] Mrinal Haloi. Traffic sign classification using deep inception based convolutional networks. CoRR, abs/1511.02992, 2015.

[3] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. CoRR, abs/1506.02025, 2015.

[4] A. Arcos-Garcia, J. Alvarez-Garcia, and Luis M.Soria-Morillo. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. In Neural Networks, pages 158–165, 2018.