Khalil Fadhel
Udacity- Self Driving Car Course
Term 1

23 July 2017

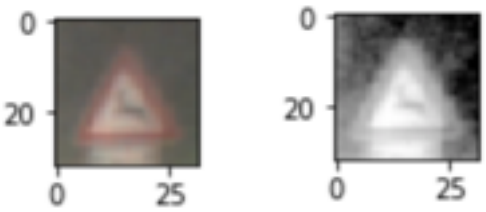
Traffic Sign Classification

In this project we aim to classify the German road traffic signs. We use tensorfzlow as our deep learning framework to train our convolutional neural network CNN and python as the scripting language. Before training the model we start by processing the dataset images using bumpy and openCV.

After evaluating the network on the test dataset, we test our model's performance on traffic sign web images. We visualise the state of the network in different steps before the general conclusion.

Pre-Processing Images

We start by transforming the input images to grayscale. As the traffic sign images features base more on the shape, this help the network train faster. Once we transformed the images to grey scale we apply an histogram equalisation using Opencv to improve the images contrast. We end the image processing by normalising the input images.and mean entering. To do so, we subtract 127.5 and dividing by 127.5 which centralise the image data in the interval [-1..1].



[Preprocessing Images]

Training Model

Once the data has been pre-processed, we start training the model. To do so, we implement LeNet architecture using Tenserflow. We use Softmax as an activation function and the cross entropy to calculate the error. We train the model over 10 epochs with a batch size of 128 and a learning rate of 0.015. We get an accuracy of 93.4% over the validation set and 90.3% over the test set.

```
Training...

EPOCH 1 ...
Validation Accuracy = 0.857

EPOCH 2 ...
Validation Accuracy = 0.880

EPOCH 3 ...
Validation Accuracy = 0.916

EPOCH 4 ...
Validation Accuracy = 0.910

EPOCH 5 ...
Validation Accuracy = 0.917

EPOCH 6 ...
Validation Accuracy = 0.934

EPOCH 7 ...
Validation Accuracy = 0.930

EPOCH 8 ...
Validation Accuracy = 0.936

EPOCH 9 ...
Validation Accuracy = 0.943

EPOCH 10 ...
Validation Accuracy = 0.934

Model saved
INFO:tensorflow:Restoring parameters from ./lenet
Test Accuracy = 0.903
```

[Training model using LeNet]

Testing on Web Images

We download 5 German traffic signs from the internet to test our network on images different from the ones it was trained on. We prepare the images by cropping and rising to 32*32*3, the input shape of our network. We pass our images through the same preprocessing tunnel as our training images. Once pre-processed we pass our web images through the network and calculate the accuracy of the predictions. We succeed to get 4 of 5 images' prediction right.

[Testing model on web images]

```
INFO:tensorflow:Restoring parameters from ./lenet
Test Accuracy = 0.800
```

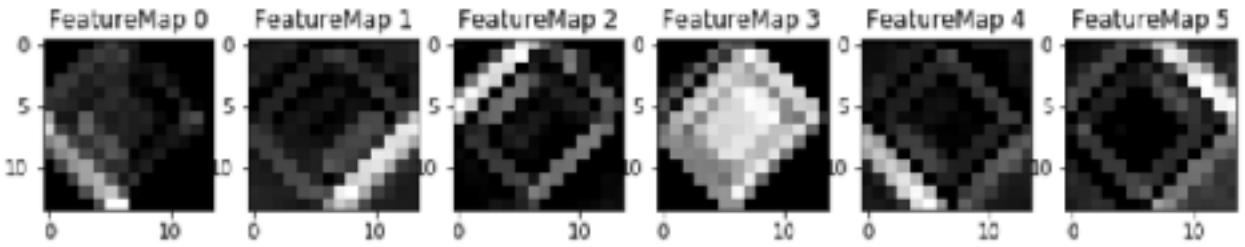
[Test accuracy on web images]

```
TopKV2(values=array([[  9.99989748e-01,   5.58895772e-06,   3.79960125e-06,
          8.72195585e-07,   2.20896776e-08],
       [  9.99208152e-01,   7.59772200e-04,   2.32797775e-05,
          8.80266634e-06,   6.50256879e-08],
       [  9.99996185e-01,   3.42940689e-06,   3.70464591e-07,
          3.46833637e-08,   2.11957314e-08],
       [  9.27322268e-01,   7.06416070e-02,   6.40991319e-04,
          5.88020484e-04,   2.96846381e-04],
       [  9.99250472e-01,   7.34325789e-04,   7.07520845e-05,
          3.69515897e-06,   1.68994495e-06]], dtype=float32), indices=array([[11, 18, 21, 40, 27],
       [18, 37,  0, 40, 11],
       [12, 42,  6,  7, 40],
       [17, 14, 38,  1, 40],
       [13, 35, 33, 12, 39]], dtype=int32))
```

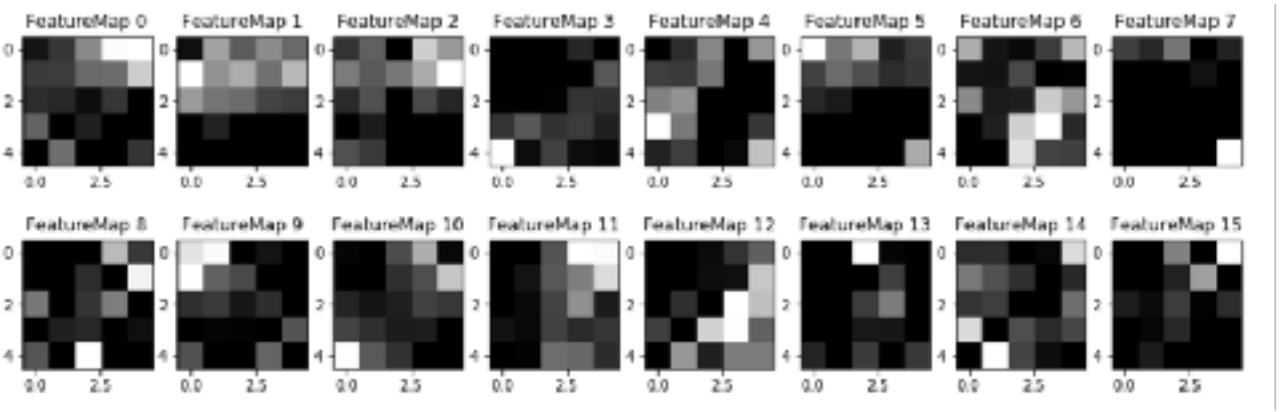
[Top 5 Softmax Probabilities For Each Web Image]

Visualise CNN State

We finalise by getting the state of our images in different state of the network. We visualise below the state of an image of the priority road after passing through the first and second convolution layers.



[Features Map of the first convolution layer]



[Features Map of the second convolution layer]

Conclusion

In this project we were able to use Tensorflow to build and train a network that classifies the German Traffic Signs. We succeed the optimise our network to a test accuracy of 90%. We can

improve further the network by enhancing the image pre-processing: reducing distort, mask circle and triangle shapes… We can also experiment with other CNN architectures and parameters.