Traffic Sign Classification with Deep NNs

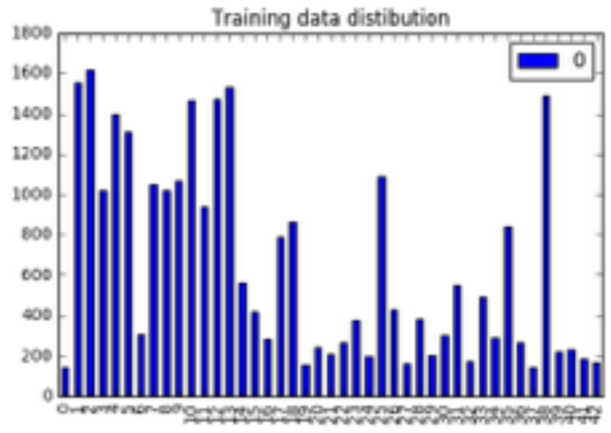
This report describes my approach on training NNs for the classification of

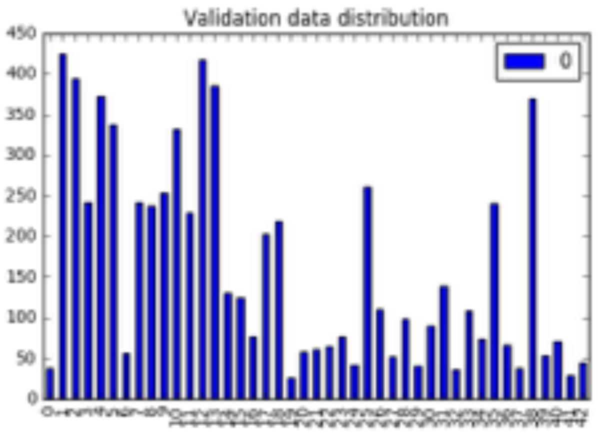
1. Data Interpretation

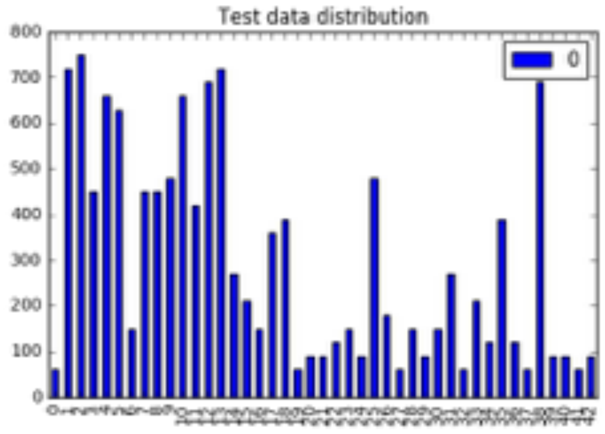
The data used was split into the following datasets

```
Number of training examples = 23315
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```

As a visual exploration I have used bar plots that identified the most common traffic signs. As it turns out, the training, test and validation sets contain similar distributions w.r.t. the inputs. This implies an good conditions for the model validation.







A numeric representation of the most common classes in the data sets is listed below:

```
Most common classes training:
(2, 1616)
(1, 1555)
(13, 1534)
(38, 1490)
(12, 1473)
(10, 1468)
(4, 1397)
(5, 1312)
(25, 1089)
(9, 1067)


 Most common classes validation:
(1, 425)
(12, 417)
(2, 394)
(13, 386)
(4, 373)
(38, 370)
(5, 338)
(10, 332)
(25, 261)
(9, 253)


Most common classes testing:
(2, 750)
(1, 720)
(13, 720)
(12, 690)
(38, 690)
(4, 660)
(10, 660)
(5, 630)
(9, 480)
(25, 480)
```

2. Data Preprocessing

The provided data contained color information. Since, however, the classification is concerned with identifying edges and shapes rather than the colour, I have simply removed the colour information by applying the np.mean function over the entire datasets (a simple way of grayscaling). This reduced the input depth from 3 to 1 and implied computational benefits.

3. Deep NN Design

As the provided data implied more complex shapes and edges compared to letters (the intended use of LeNet), I have taken the LeNet example and extended it. My approach was to “treat” the data a bit more gently by removing the first pooling operation from the LeNet example. Moreover, I have added one more convolutional layer and reduced the filter sizes in order for my model to be able to identify finer shapes.
My last change was to replace the second max pooling operation with an average pool.

My final design

1. Convolutional Layer with size 5x5x6 (based on my observations the selected size in the LeNet lab for the first layer was good fit as smaller and bigger filters had a negative result)
2. My second layer is a 3x3 convolution that had identical size of k (6) compared to the first layer
3. The third layer is a reduced 2. Layer with size 2x2.

In all layers above, I have assigned the k-size to 6 as this was a tradeoff between computational performance and results.
4. Average Pooling Layer with a stride of 2x2 and a filter size of 2x2.
My assumption for the selection of the parameters: since the previous layers should remain very detailed information, a mean over a broader area should lead to loss of features.

The remaining layers are fully connected while the output sizes were chosen to be around 1/3 of the input size.

4. Results

For the LeNet lab solution I have received an accuracy of Validation Accuracy = 0.952 for the validation set and 0.857 for the test set using the learning rate of 0.001 and 10 epochs.
Both values have been selected based on my observations and the fact that higher values for the learning rate led to poor results and higher amount of epochs to overfitting.

Using my “modified” LeNet architecture, I have received a validation accuracy of 0.979 and a test accuracy of 0.911.

My approach of “gently” treating the data was therefore successful.

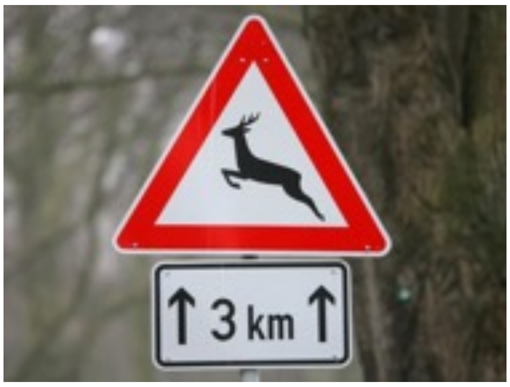
Note: I have run tests with a 1x1 Convolutional Layer. These implied very poor results in classifying completely new images (refer below). My assumption is that the 1x1 Layer represents a very drastic approach in identifying certain shapes found on traffic signs.

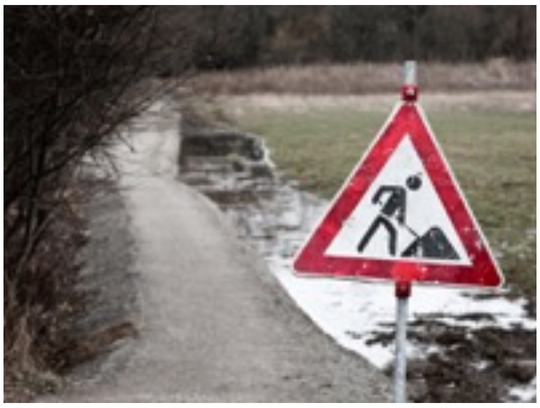
5. Applying new data

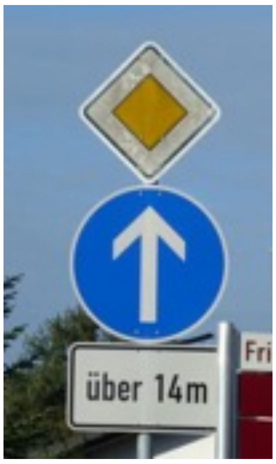
In order to check if my model was “fit” enough to classify other data than the one used for training and validation, I have selected two different data sets: “easy” and “hard”

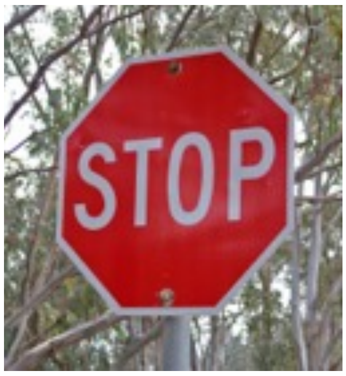
1. The “easy” dataset contains 5 straightforward images, taken under fairly good attributes w.r.t. illumination perspective, that should be no problem for my model.

2. The “hard” dataset contained one image of the “easy” class and 4 images that are fairly difficult to interpret as these include multiple roadsigns and / or the environment.









Note: since images found on the web imply different sizes, I have preprocessed these using PIL.

For the “easy” data, the prediction had a 100% accuracy (80% in case of LeNet).

The resulting top 3 softmax probabilities also implied almost no error in the “easy” dataset

Stop sign:
```
TopKV2(values=array([[0.999903, 0.000092, 0.000005]],
dtype=float32), indices=array([[14,  1,  2]],
dtype=int32))
```

Forward only:
```
TopKV2(values=array([[1.000000, 0.000000, 0.000000]],
dtype=float32), indices=array([[35,  3, 19]],
dtype=int32))
```

Road work:
```
TopKV2(values=array([[1.000000, 0.000000, 0.000000]],
dtype=float32), indices=array([[25, 28, 30]],
dtype=int32))
```

Priority road:
```
TopKV2(values=array([[1.000000, 0.000000, 0.000000]],
dtype=float32), indices=array([[12, 40, 16]],
dtype=int32))
```

Wild animals crossing:
```
TopKV2(values=array([[0.869449, 0.130551, 0.000000]],
dtype=float32), indices=array([[31, 25, 11]],
dtype=int32))
```

For the “hard” data however, only the “forward only” image that included two more traffic signs was identified correctly (apart from the easy stop sign)

I have experienced the same results with the unmodified LeNet architecture.

The top 3 softmax probabilities are listed below:
It seems that my model was misinterpreting the traffic signs with a high confidence.

Stop sign (correct):

```
TopKV2(values=array([[0.999903, 0.000092, 0.000005]],
dtype=float32), indices=array([[14,  1,  2]],
dtype=int32))
```

Forward only (incorrect classification of 31, Wild animals crossing):

```
TopKV2(values=array([[0.994177, 0.005436, 0.000174]],
dtype=float32), indices=array([[31, 35, 20]],
dtype=int32))
```

Road work (incorrect classification of 6,End of speed limit (80km/h)):

```
TopKV2(values=array([[0.974730, 0.025270, 0.000000]],
dtype=float32), indices=array([[6, 1, 0]], dtype=int32))
```

Priority road (incorrect classification of 1,Speed limit (30km/h)):

```
TopKV2(values=array([[1.000000, 0.000000, 0.000000]],
dtype=float32), indices=array([[ 1, 40, 17]],
dtype=int32))
```

Wild animals crossing (incorrect classification of 0,Speed limit (20km/h)):

```
TopKV2(values=array([[0.999337, 0.000659, 0.000003]],
dtype=float32), indices=array([[ 0, 29,  8]],
dtype=int32))
```