Traffic Sign Project Report

March 22, 2018

1 Project Goal

The goal of this project is to build a convolutional neural network to classify traffic signs in the German Traffic Sign dataset[1], and evaluate its performance.

2 Data Exploration

The dataset consists of three subsets.

	Training	Validation	Test
Example Count	34799	4410	12630

The images in the dataset are in 32x32x3 RGB format. The total number of sign classes is 43.

We first verify that the class label distribution in each subset are consistent. See Figure 1 for a bar chart of class label count per subset. As we can see, the classes are not distributed evenly, which might cause some labels to be predicted more accurately later on, but the distribution is consistent in all three subsets, so evaluation with the validation and test sets should properly reflect model performance.

We also check if the colour distribution is consistent in three subsets. in Figure 2 we see that the RGB distribution is consistent in all three datasets.

3 Image Preprocessing

Before we train our network on the images, we first need to preprocess the data to provide a normalised input for the network. the easiest approach is to subtract the mean from the images half the range, but we observed that some images have larger range of colour and brightness variations than others, therefore a per-image normalisation is used to ensure consistent variation ranges for all images.

We also perform a data augmentation step for all training data by adding some random noise to the images. During training, before the network consumes the input, the input images are randomly perturbed in hue, brightness, saturation and contrast by a small amount. This helps increase the robustness of the network and reduce overfitting. We do not expand the original training set. Instead, we perform this perturbation each passthrough so that each epoch, the network trains on slightly different versions of each training example. Luckily, we can incorporate this step into the Tensorflow computation graph.

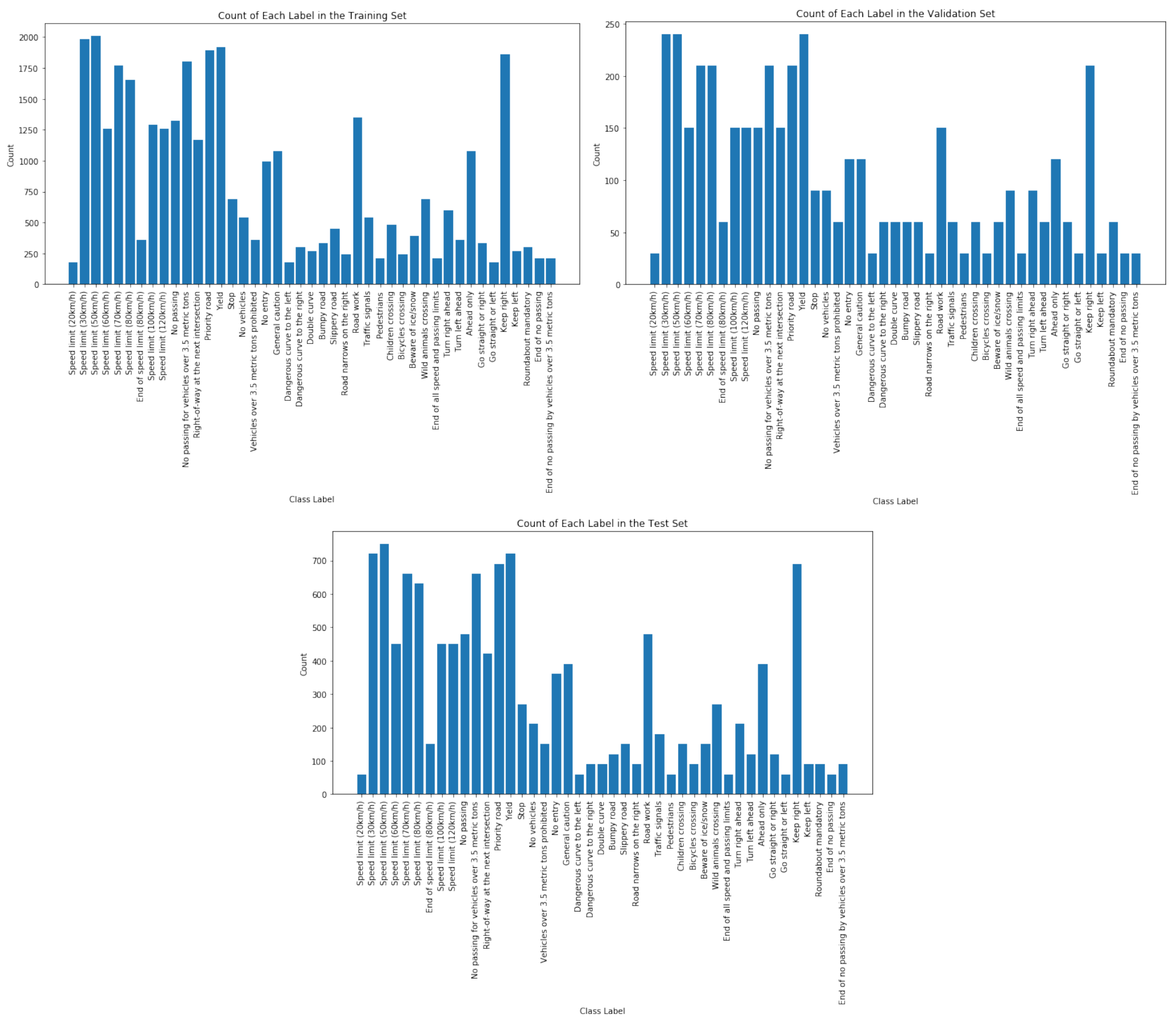

Figure 1: Distribution of Class Labels in Each Subset

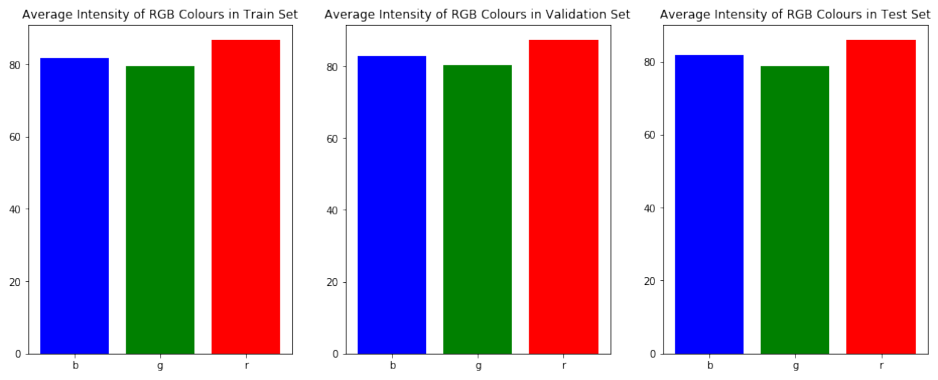

Figure 2: Distribution of RGB colours in Each Subset

4 Model Architecture

Our network uses the following architecture:

Layer No.	Layer Type	Dimensionality	Activation	Note
1	Conv2D	16x[3, 3]	ReLU	Valid Padding, L1 Regularisation (0.1)
2	MaxPooling	[2, 2]	None	Stride=2
3	Conv2D	32x[3, 3]	ReLU	Valid Padding, L1 Regularisation (0.1)
4	MaxPooling	[2, 2]	None	Stride=2
5	Dense	512	ReLU	Batch Normalisation
6	Dropout	-	None	Drop Rate=0.7
7	Dense	512	ReLU	Batch Normalisation
8	Dropout	-	None	Drop Rate=0.7
9	Dense	43	Linear / Softmax	Output

The computation graph has an auxiliary boolean input *is_training* to inform the network whether to perform noise adding and dropout in forward passthrough. The whole forward pass architecture can be seen in Appendix Figure 10.

[3, 3] kernels are found to perform better than [5, 5] kernels, likely due to the fact that the details in traffic signs exhibit difference on a small scale. L1 regularisation here is used to induce sparsity in the convolution kernels so they focus on different aspects of the input image. A pair high dimensional dense layer is used in conjunction with a high dropout rate to achieve a balance between expressiveness and overfitting control.

5 Training and Evaluation

The network is trained on the training dataset with a batch size of 32 over 20 epochs. (When training on a GPU, the batch size can be reduced to limit VRAM usage.) Input images are shuffled each epoch. Loss and accuracy are evaluated per epoch on both the training and validation set. The training vs validation loss over training steps is shown in figure 3.

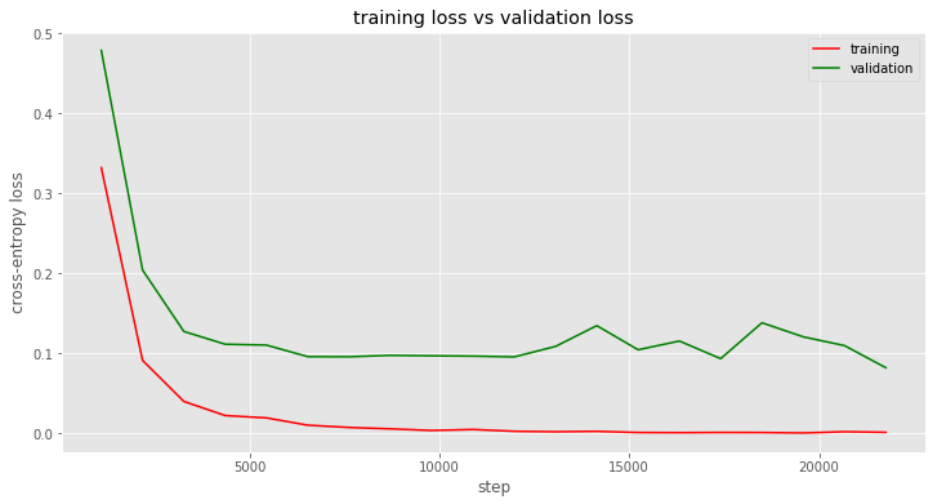

Figure 3: training vs validation loss

The network achieves **96.22%** accuracy on the test dataset.

6 Performance Analysis

To analyse the performance of the network in more detail, we acquired five new German traffic sign images from Google Image Search and predicted their top 5 labels with the classifier. Here are the results:

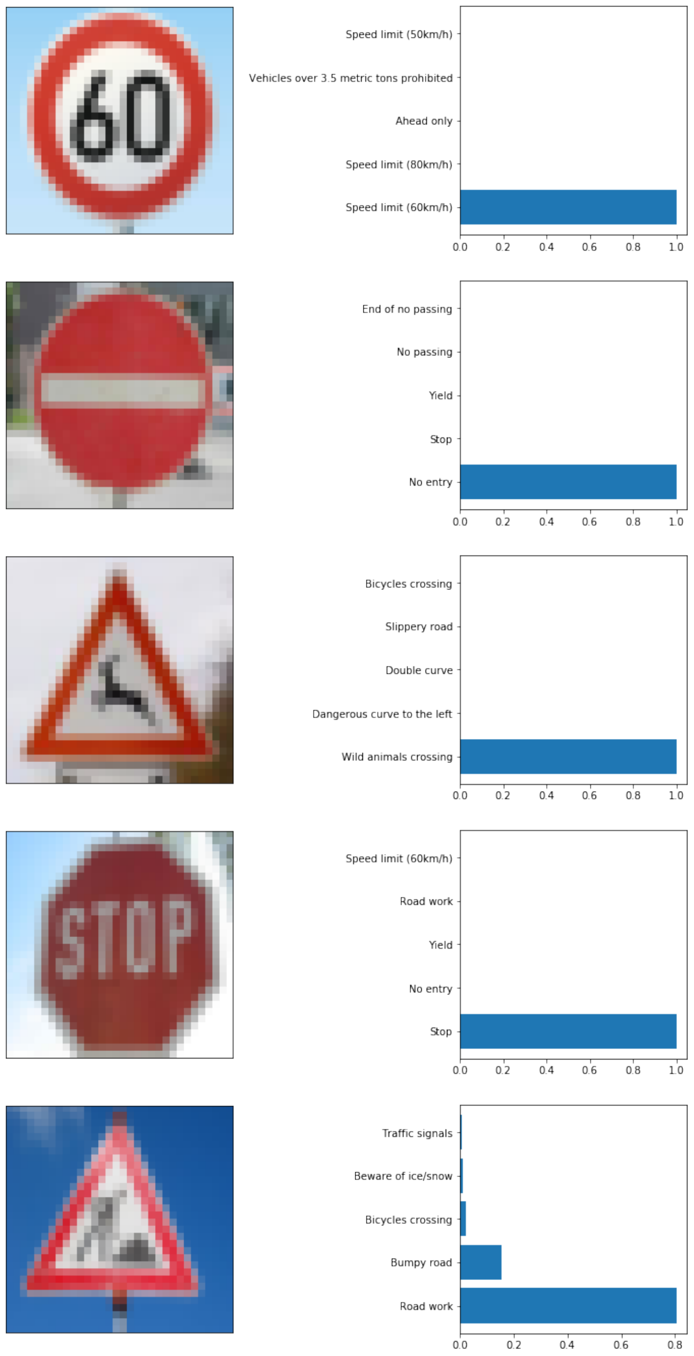

Figure 4: Prediction with New Images

As we see, the network is able to provide correct predictions for all images. In addition, it is able to provide "confident" predictions for all images except "Road Work". However, we will show later that the softmax probability is not a good measure of the network's confidence.

Since the network is able to correctly classify all new images, we have to examine the misclassified images in the test dataset to figure out how it behaves when it fails to classify correctly. Here are some of the examples of misclassification:

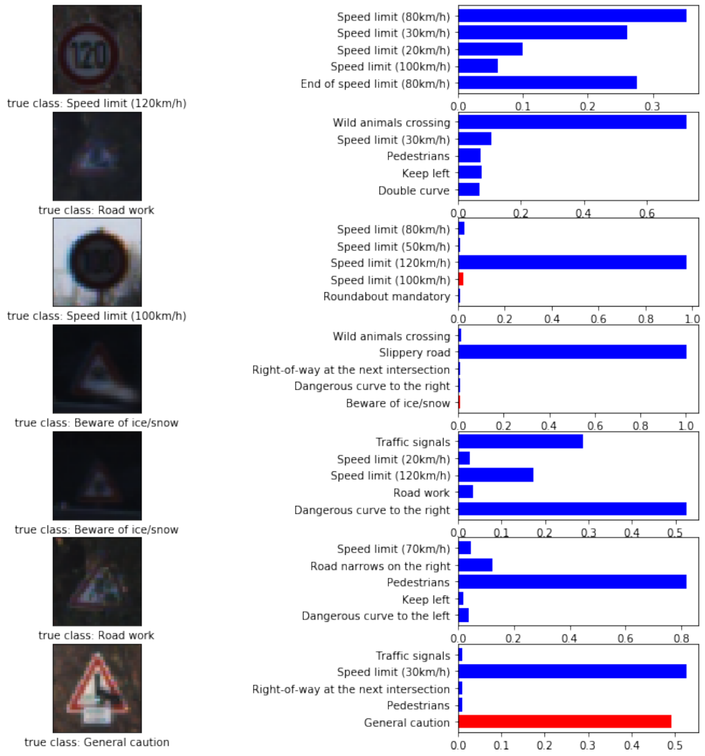

Figure 5: Misclassifications

Here, we see that in some cases, the softmax probability reflects the uncertainty of the classifier (such as the first image), whereas in some other cases, even though the image is so blurry that eeven a human cannot easily label it, the network provides an extremely "confident" output that is quite wrong. Is there another way to estimate the uncertainty of the network? Recall that we do not include the noise adding and dropout steps in the evaluation pipeline. One idea is to add these stochastic elements back into the network, let it predict with uncertainty, and examine a sample of its prediction to gauge its confidence. As an example, we take the sixth image ("Road work") and run it through the network 10000 times with noise and dropout enabled (Figure 6).

As we see in the bar chart, even though the network is still unable to guess the correct label, the distribution of different labels in the sample is closer than the softmax probabilities to where the uncertainty should be.

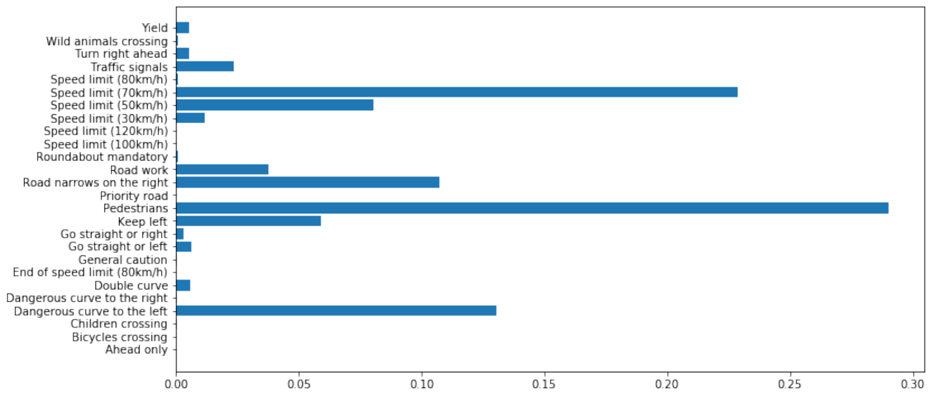

Figure 6: Stochastic Prediction Sample

We also calculate the per-class percision and recall in Appendix Table 1. The classes with the worst precision are **Speed limit (20km/h), Pedestrians and Beware of ice/snow**, and the classes with the worst recall are **Speed limit (120km/h), Pedestrians and Dangerous curve to the right**. Overall, the network is struggling the most with the class **Pedestrians**. We wonder why this is the case. Here are the classes that **Pedestrian** images are most commonly classified as:

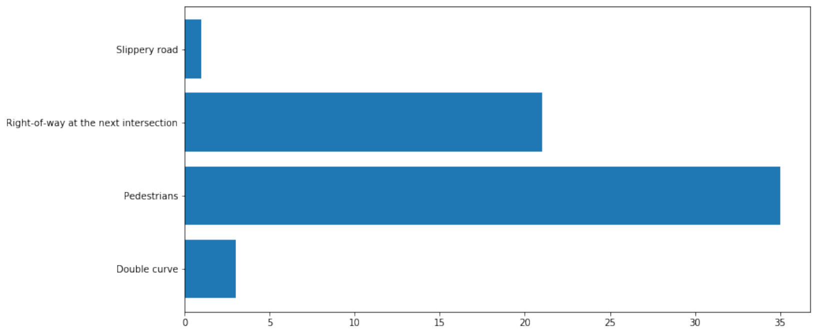

Figure 7: Pedestrian Class Prediction Outcome

It appears that the network frequently confuses **Pedestrian** with **Right-of-way at the next intersection**, which is indeed very similar in appearance. At the given image resolution, it is even difficult for humans to distinguish between the two.

7 Activation Visualisation

Neural networks are often treated as black boxes, as the precise function of each layer and each node is difficult to pinpoint. However, sometimes it is possible to crudely estimate the function of a convolution kernel in a CNN by examining its activation patterns. Here, we visualise the mean activations of our convolutional layers for all circular-plate signs versus triangular-plate signs. We focus on the first convolutional layer, as the second layer is harder to interpret.

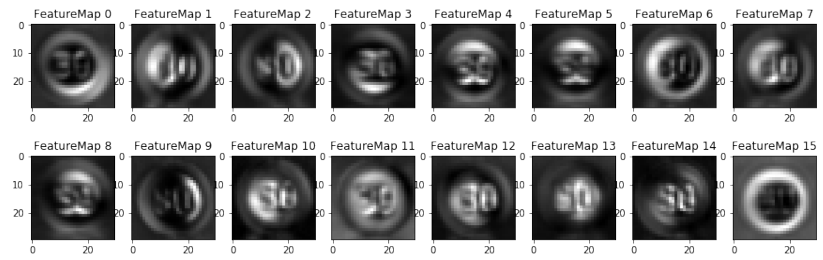
Figure 8: Conv Layer 1 for Circular Plates

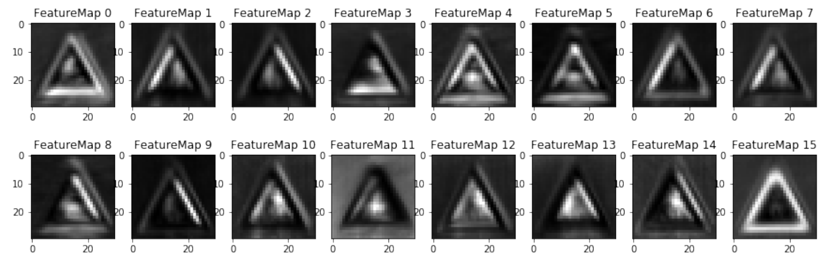
Figure 9: Conv Layer 1 for Triangular Plates

Notice that in both cases, the feature map 15 in the first convolution layer is always detecting the coloured stripe / ring around the sign, regardless of the shape of the sign, whereas feature map 13 in the first convolution layer always focuses on the interior of the sign. The different kernels focus on different aspects of the input images and create more meaningful features for the following layers.

8 Further Improvements

The current network architecture and data pipeline are not perfect and may still be improved. For example, the data augmentation step only used per-pixel noise-adding. By utilising other transformations such as rotation and translation, we may further improve the performance. The network also appears to struggle a little bit with some off-centre images or smaller signs, so some form of multi-scale solution (such as parallel convolutional layers of different kernel sizes) can be helpful. Also, the training curve of the network shows that the validation loss has not yet stablised or "bounced back", so we may not have reached optimality during training yet. By tweaking the training parameters, we may be able to reach a better optimal point.

References

[1] Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks* (2011), pp. 1453–1460.

Appendix

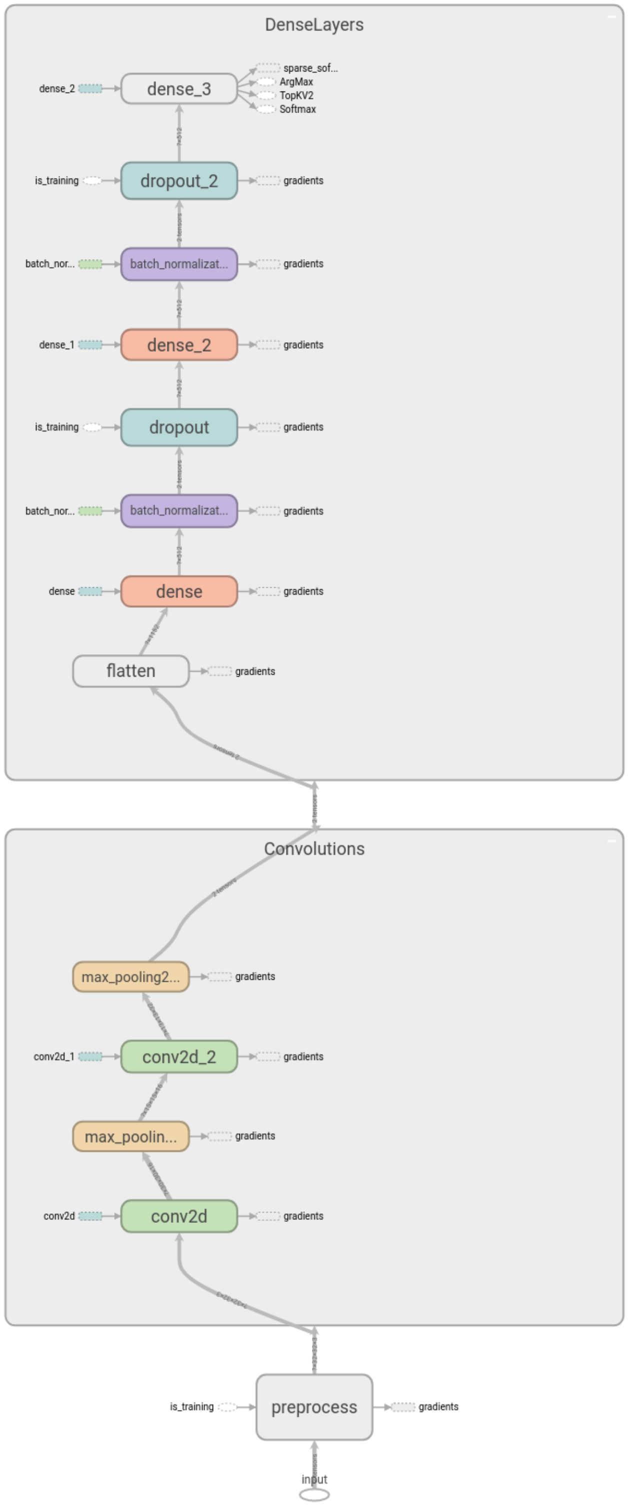

Figure 10: Network Architecture

Table 1: Per-class Precision and Recall

row	label_ID	true_label	precision	recall	precision_rank	recall_rank
0	0	Speed limit (20km/h)	0.566667	1.000000	42	0
1	1	Speed limit (30km/h)	0.988889	0.966079	15	26
2	2	Speed limit (50km/h)	0.980000	0.965834	18	27
3	3	Speed limit (60km/h)	0.935556	0.972286	32	24
4	4	Speed limit (70km/h)	0.966667	0.990683	23	12
5	5	Speed limit (80km/h)	0.953968	0.909228	28	35
6	6	End of speed limit (80km/h)	0.820000	0.991935	38	10
7	7	Speed limit (100km/h)	0.882222	0.985112	35	15
8	8	Speed limit (120km/h)	0.993333	0.809783	8	42
9	9	No passing	0.987500	0.979339	16	22
10	10	No passing for vehicles over 3.5 metric tons	0.975758	0.993827	22	7
11	11	Right-of-way at the next intersection	0.992857	0.892934	9	38
12	12	Priority road	0.995652	0.984241	5	16
13	13	Yield	0.997222	0.993084	3	8
14	14	Stop	0.996296	1.000000	4	3
15	15	No vehicles	1.000000	0.981308	2	20
16	16	Vehicles over 3.5 metric tons prohibited	0.993333	1.000000	7	2
17	17	No entry	0.991667	1.000000	12	1
18	18	General caution	0.935897	0.973333	31	23
19	19	Dangerous curve to the left	1.000000	0.983607	1	17
20	20	Dangerous curve to the right	0.977778	0.814815	20	40
21	21	Double curve	0.855556	0.905882	36	36
22	22	Bumpy road	0.950000	0.957983	30	28
23	23	Slippery road	0.980000	0.942308	19	33
24	24	Road narrows on the right	0.900000	0.843750	34	39
25	25	Road work	0.954167	0.950207	27	31
26	26	Traffic signals	0.850000	0.987097	37	14
27	27	Pedestrians	0.583333	0.813953	41	41
28	28	Children crossing	0.953333	0.966216	29	25
29	29	Bicycles crossing	0.933333	0.893617	33	37
30	30	Beware of ice/snow	0.720000	0.990826	40	11
31	31	Wild animals crossing	0.966667	0.981203	24	21
32	32	End of all speed and passing limits	1.000000	0.937500	0	34
33	33	Turn right ahead	0.995238	0.995238	6	6
34	34	Turn left ahead	0.991667	0.952000	11	30
35	35	Ahead only	0.992308	0.992308	10	9
36	36	Go straight or right	0.966667	0.983051	25	19
37	37	Go straight or left	0.983333	0.983333	17	18
38	38	Keep right	0.989855	0.997080	13	5
39	39	Keep left	0.977778	0.956522	21	29
40	40	Roundabout mandatory	0.966667	0.988636	26	13
41	41	End of no passing	0.816667	1.000000	39	4
42	42	End of no passing by vehicles over 3.5 metric ...	0.988889	0.946809	14	32