Traffic Sign Classifier

Goals

The goals / steps of this project are the following:

- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Summary of Dataset

The size of the dataset was calculated using numpy library.

- Number of training examples = 34799
- Number of testing examples = 12630
- Image data shape = (32, 32, 3)
- Number of unique image classes/labels = 43

Below is the collage of sample images from the dataset:

Fig: Collage of images from dataset

And the number of images per class is displayed in the chart below:

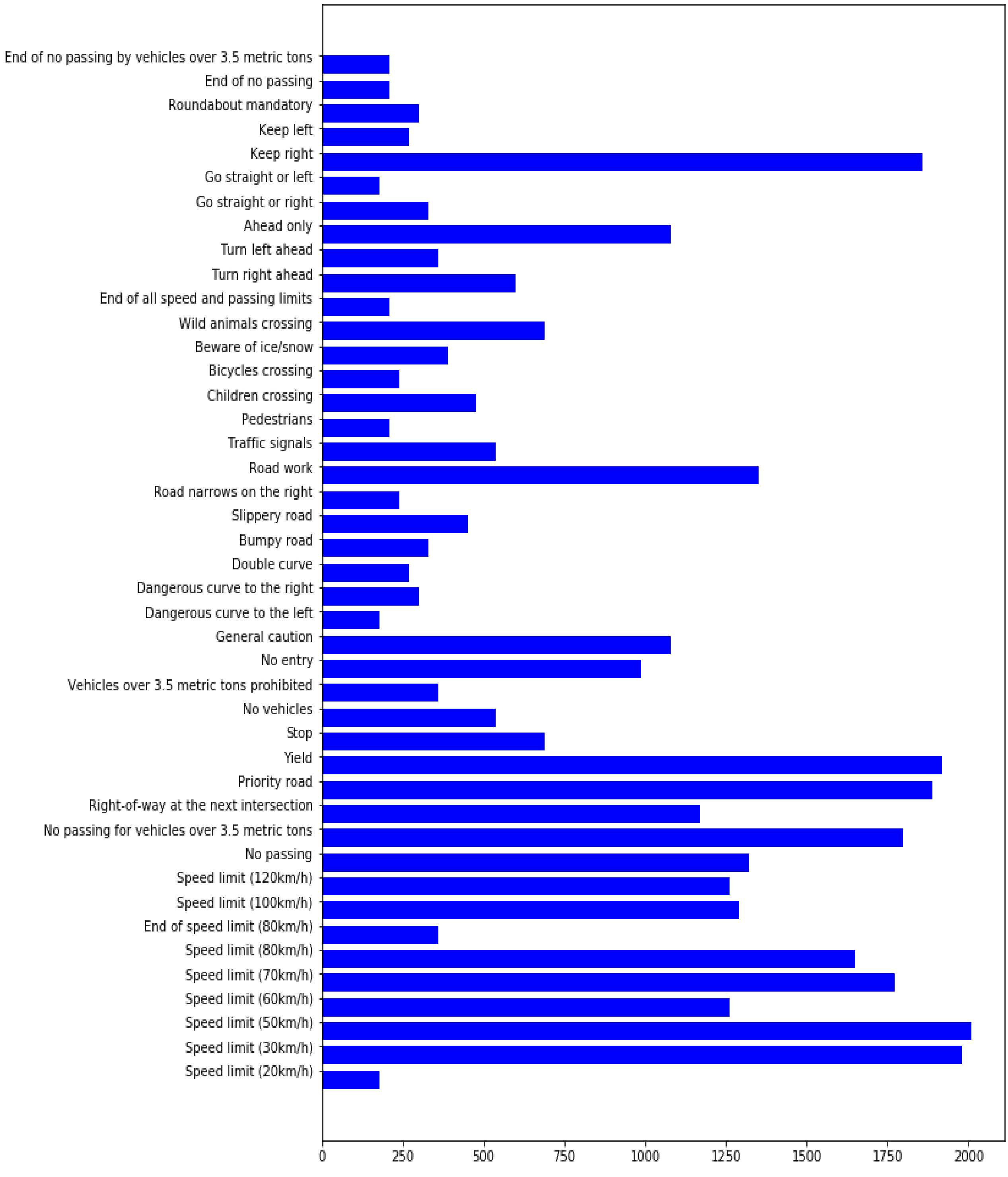


Fig: Distribution of images across classes in original dataset

Augmenting Dataset

It can be seen that the number of images per class in unbalanced. This will result in the model being trained to classify labels with more number of images more accurately than the labels with less number of images. To overcome this problem, the dataset was augmented with rotated and noisy images in such a way that there were at least 1500 images of each type. This ensures that there are ample number of images per classes and the model has good prediction accuracy for all the classes.

Rotated image from augmented training set:

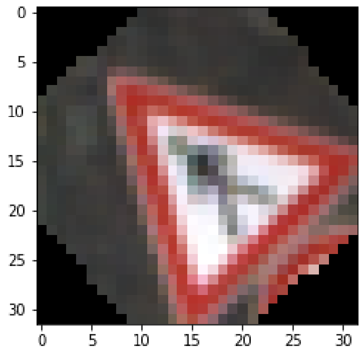


Fig: Rotated traffic sign: pedestrians

Next, the dataset size was further increased by adding random noise to each image in the original dataset and appending the noisy image to the augmented dataset. After augmenting rotated and noisy images, the training dataset had 130317 images.

Noisy image from augmented training set:

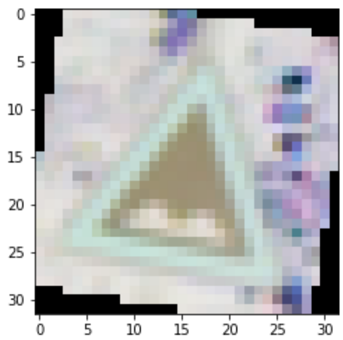


Fig: Noisy traffic sign image: bumpy road

The image distribution in the augmented dataset is displayed below.

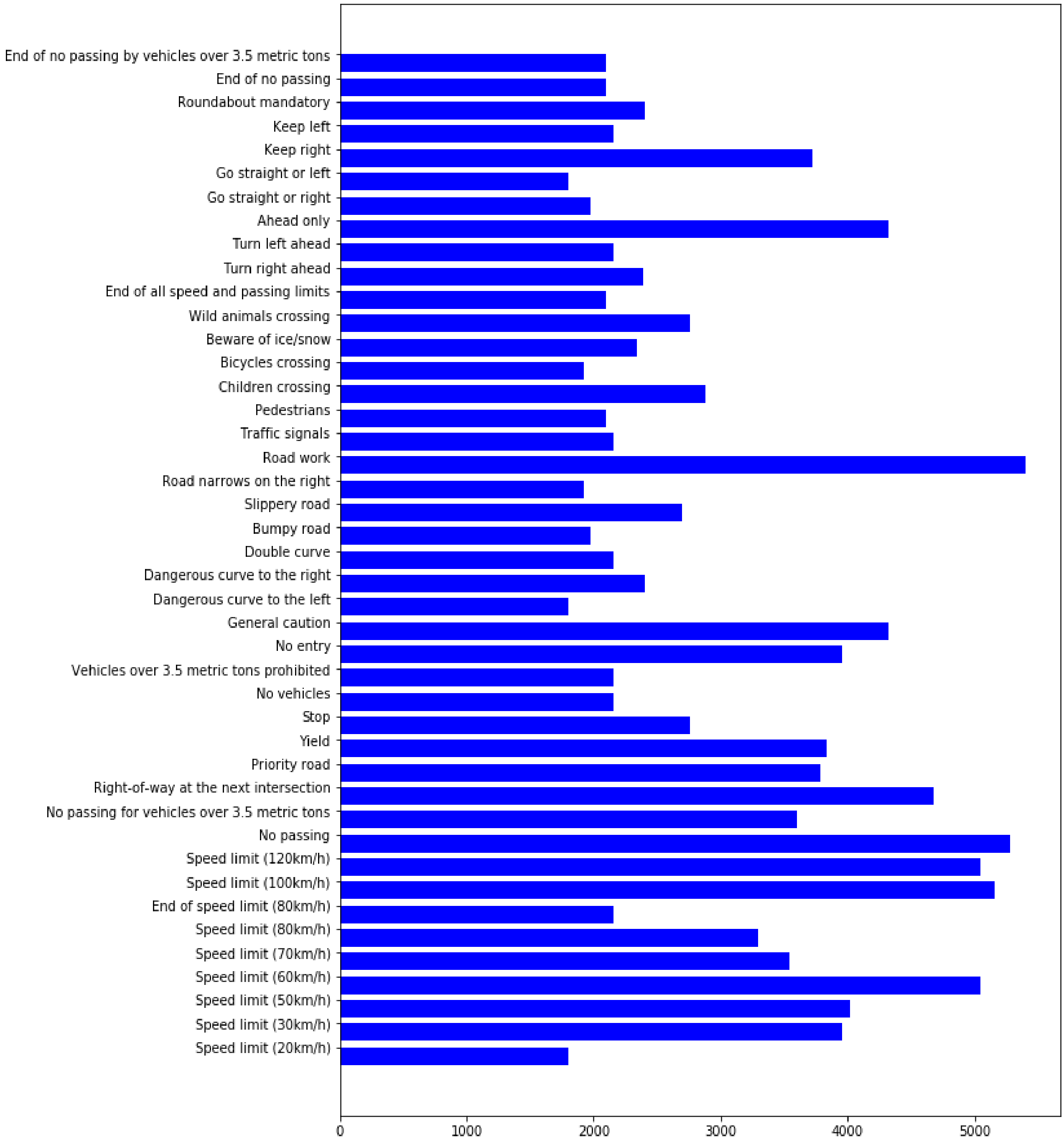


Fig: Distribution of images across classes in augmented dataset

Normalize Images

After augmenting the dataset, the images in the dataset were normalized using numpy library. The value of each pixel in the image was replaced with (pixel - 128)/128. This will convert the value range of each pixel from 0 to 255 to the range of -1 to 1. Normalization is done to facilitate Stochastic Gradient Descent algorithm to reach minimum loss for weights and bias faster. Also, normalizing the images ensures that the range of distribution of the pixels is uniform.

Model Architecture

The model used for this classifier is a modified version of LeNet5, in which, the model takes RGB images of size 32x32 as input. The model architecture consist of 8 layers:

Layer	**Description**
Input	RGB Image of size 32x32x3
Convolution	Input: 32x32x3, Output: 28x28x6
MaxPool	Input: 28x28x6, Output: 14x14x6
Convolution	Input: 14x14x6, Output: 10x10x16
MaxPool	Input: 10x10x16, Output: 5x5x16
Fully Connected Layer	Input: 400, Output: 120
Fully Connected Layer	Input: 120, Output: 84
Fully Connected Layer	Input: 84, Output: 43

Model Training

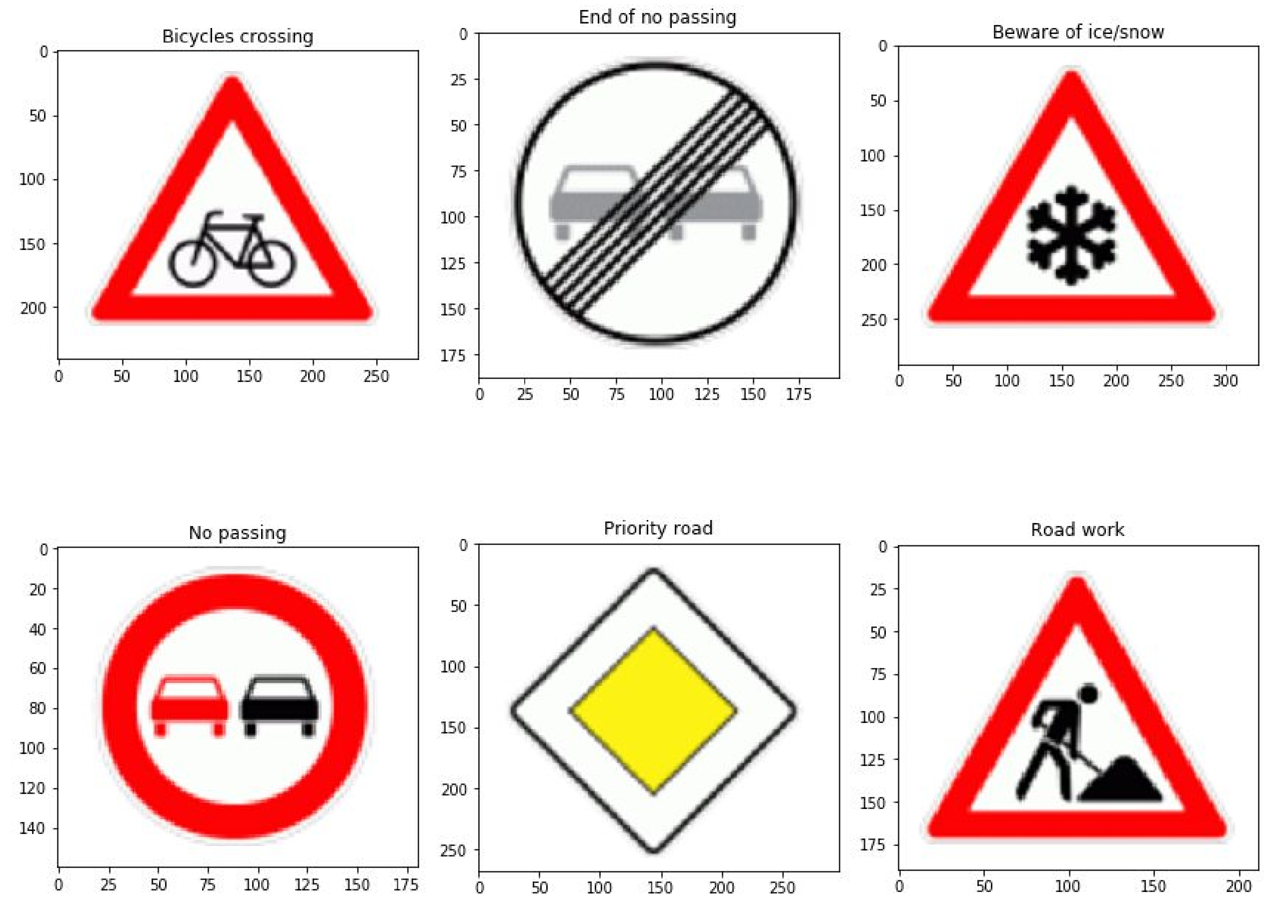
Final Hyper-parameters
==================
Epochs : 30
Batch Size: 128
Training Rate: 0.001
Optimizer: Adam-optimizer

Adam Optimizer was used to reduce loss during training. The loss was calculated as a mean of cross entropy of softmax of logits and one-hot encoded training labels. After training, with the default German Traffic Sign dataset, the trained model achieved the accuracy of 89% (with 10 epochs and batch size of 128) on the validation set. Several methods were tried to increase the accuracy. First, the training images were converted to grayscale and the default LeNet5 architecture was used. However, the validation accuracy was only as good or worse than the first training attempt. Next, the training images were normalized and the model was trained a accuracy was again the same as before. In the next training iteration, the number of layers were increased by adding an extra fully connected layer. (refer the table below). A new layer was added because each layer in the model is responsible for learning a new set of features in the image. So it was believed that an additional layer will pick an extra set of features to enable more accurate prediction.

Layer	Description
Input	RGB Image of size 32x32x3
Convolution	Input: 32x32x3, Output: 28x28x6
MaxPool	Input: 28x28x6, Output: 14x14x6
Convolution	Input: 14x14x6, Output: 10x10x16
MaxPool	Input: 10x10x16, Output: 5x5x16
Fully Connected Layer	Input: 400, Output: 200
Fully Connected Layer	Input: 200, Output: 100
Fully Connected Layer	Input: 100, Output: 75
Fully Connected Layer	Input: 75, Output: 43

With the new model architecture, the accuracy on the validation set increased to 91%, but it never got any better. For further improvement, the dataset was analyzed and it was found that some of the classes had very few training images compared to other. To overcome this problem, the dataset were augmented with rotated and noisy training images. The augmented dataset increased the number of training images from 34799 to 95517. After these changes, the model was trained on the augmented dataset with 30 epochs, batch size of 64. The trained model achieved accuracy of 94.58% on validation set and 92.99% on the test set.

Finally, the model was tested on a set of 6 images from internet. The images were pre-processed (resized to 32x32x3 and normalized) and fed into the model. The model predicted the labels for images with 100% accuracy.



And the top 5 softmax probabilities predicted by the model for the above images are listed below(correspond to images from left to right and top to bottom)

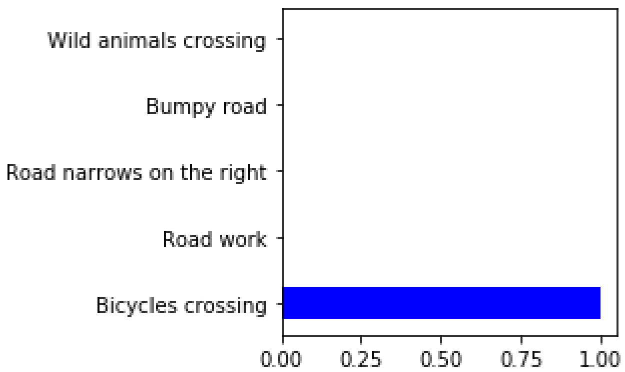


Fig 1: Bicycle Crossing

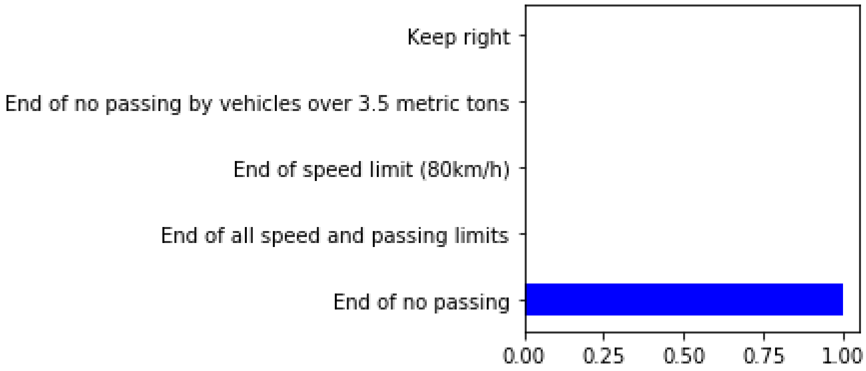


Fig 2: End of No Passing

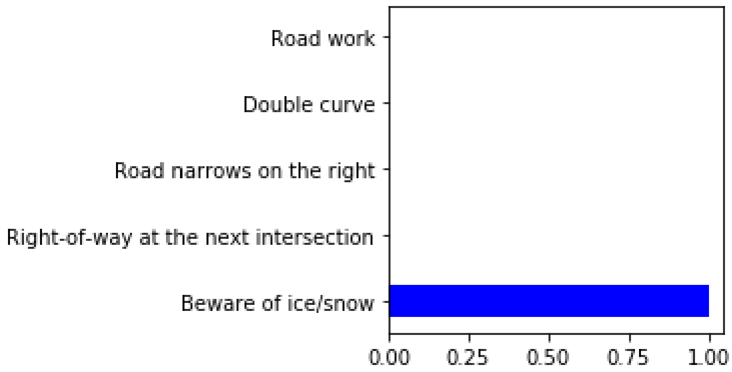


Fig 3: Beware of ice/snow

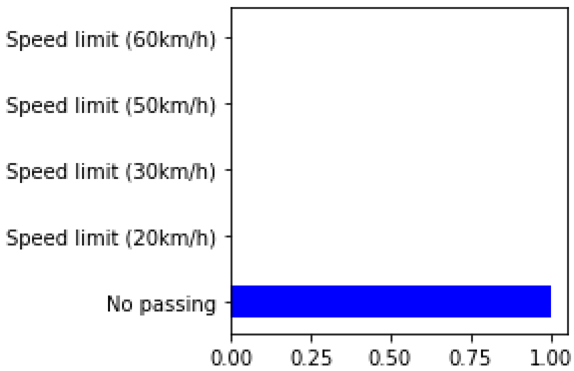


Fig 4: No Passing

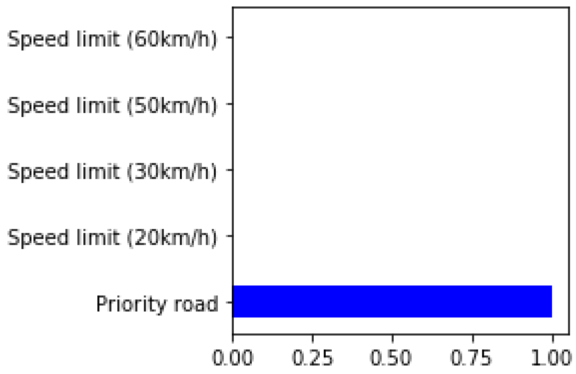


Fig 5: Priority Road

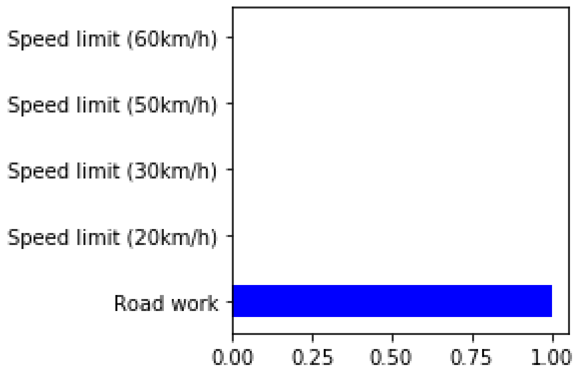


Fig 8: Road Work