Traffic Sign Recognition

Reflection Report

Introduction

The objective of this assignment was to perform traffic sign recognition on the German Traffic Signs dataset (original found here: http://benchmark.ini.rub.de/?section=gtsdb&subsection=dataset) using convolutional neural networks (CNN). We will be working with a smaller subset of it and the images are resized to a resolution of 32 x 32. We first will explore and summarise what the dataset contains, then design, train and test a CNN architecture that will classify traffic signs. We then extend this to predict the signs of five new images found on the web, then display what the probabilities of classification are for the top 5 classes, then provide conclusions.

Dataset Summary

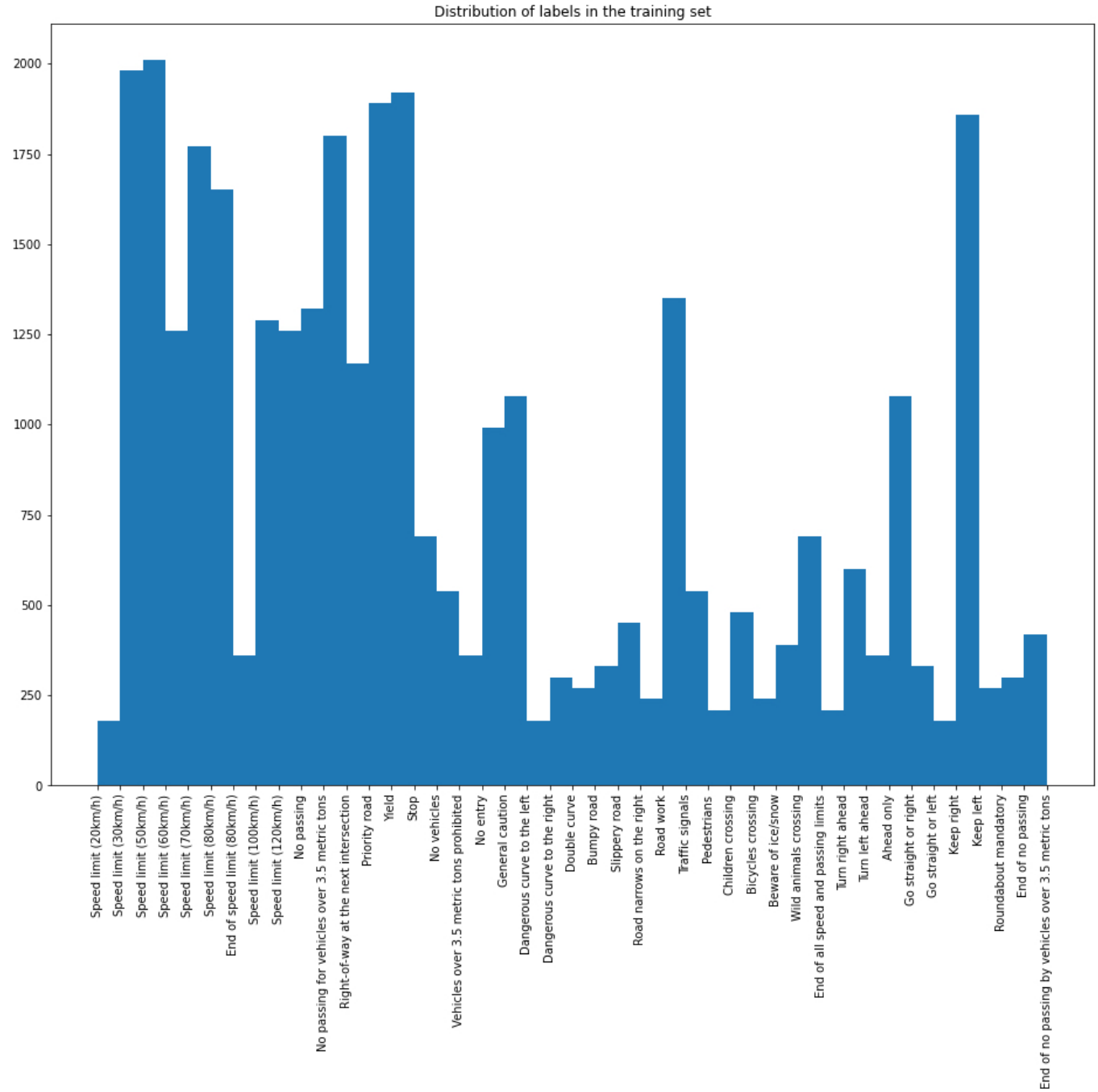
The dataset for this assignment was provided by Udacity in Python pickle files. In the notebook file for this assignment, the dataset is downloaded by running a cell which will also extract the data and put it in a local `data` directory. The data is not included with this repo for the sake of brevity but you can download the data by running the relevant cell in the notebook. The datasets fortunately are just NumPy arrays and are separated into training, validation and test datasets. Each of them are 4D arrays of images and are structured such that the first dimension is the total number of examples, the second dimension is the number of rows for an image, the third dimension is the number of columns for an image and the last dimension is the number of channels. All images are 32 rows, 32 columns with 3 channels being in RGB. For the output labels, they are simply integers ranging from 0 to 42 signifying the ID of what the sign should be detected as. The mapping from ID to the identify of the sign can be found in the `data/signnames.csv` file. You'll also have a chance to look at them when we perform the exploratory visualisation of the dataset soon.

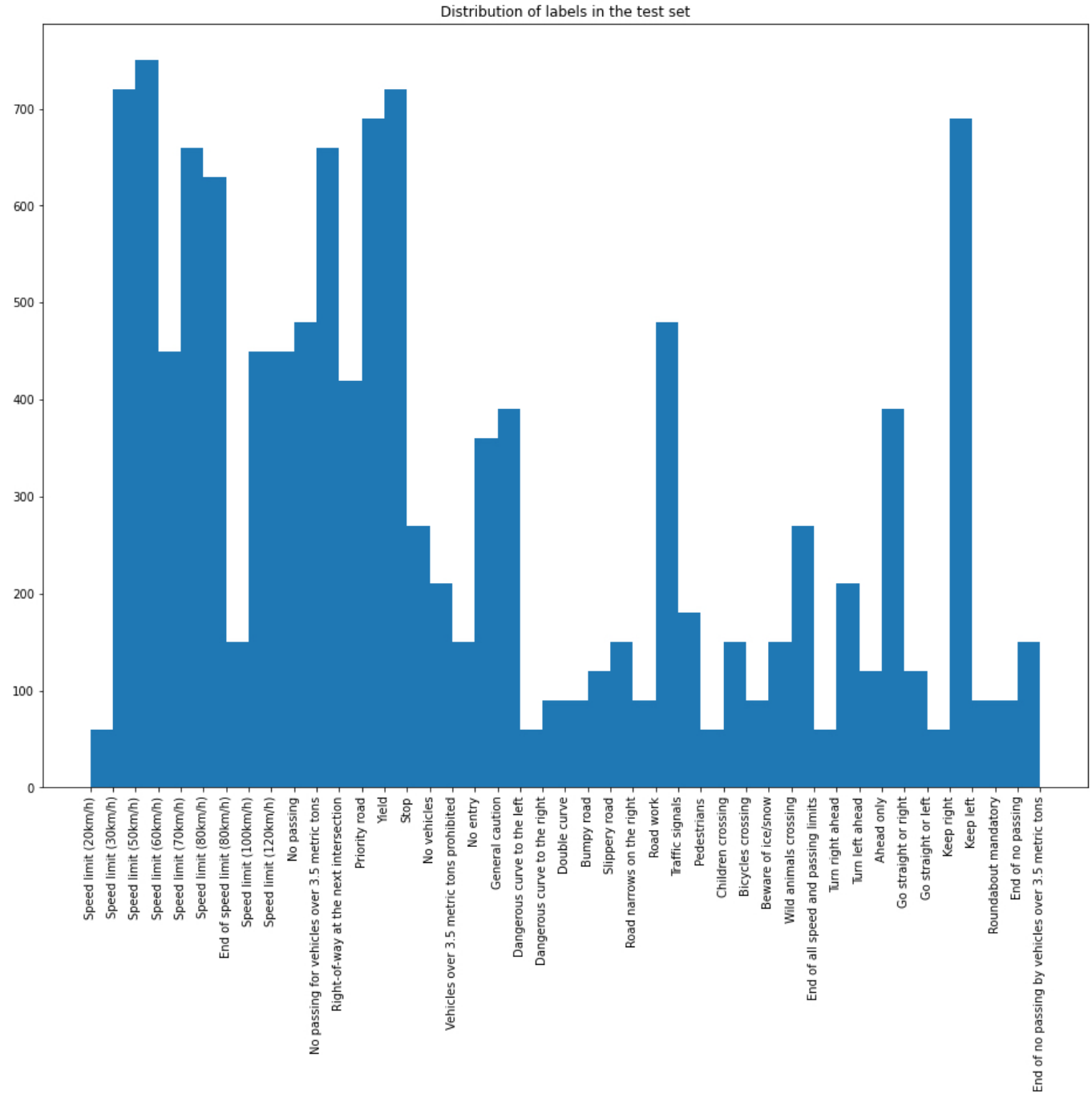
Other relevant quantities are:

- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is 32 x 32 x 3 - they are colour images
- The number of unique classes/labels in the data set is 43

Exploratory Visualisation

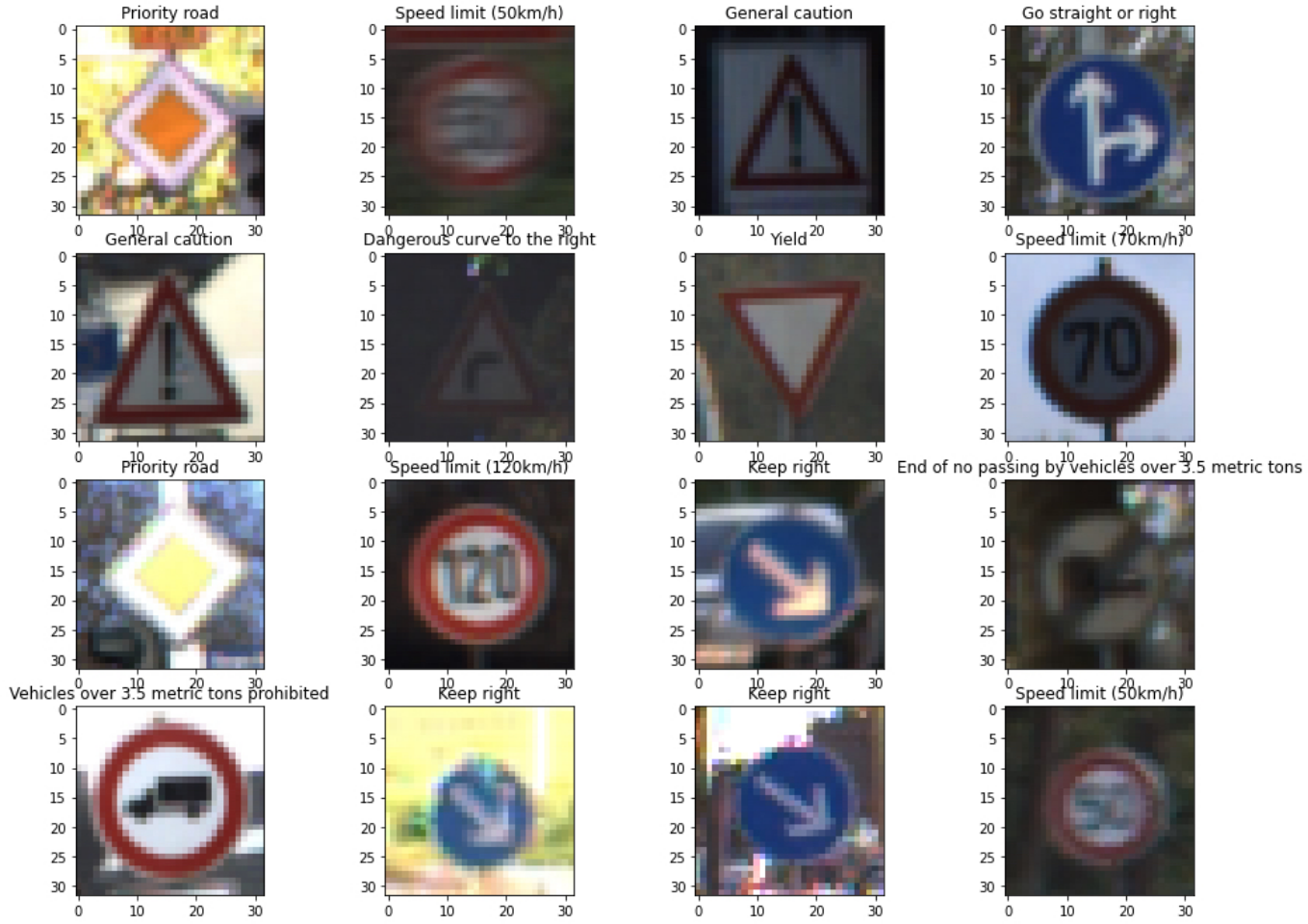
It would be interesting to have a look at the distribution of the images over the different classes to see how balanced they are. The figures below plot a histogram of the total number of images per class for each of the groups.


Distribution of labels in the training set
2000
1750
1500
1250
1000
750
500
250
0
Speed limit (20km/h)
Speed limit (30km/h)
Speed limit (50km/h)
Speed limit (60km/h)
Speed limit (70km/h)
Speed limit (80km/h)
End of speed limit (80km/h)
Speed limit (100km/h)
Speed limit (120km/h)
No passing
No passing for vehicles over 3.5 metric tons
Right-of-way at the next intersection
Priority road
Yield
Stop
No vehicles
Vehicles over 3.5 metric tons prohibited
No entry
General caution
Dangerous curve to the left
Dangerous curve to the right
Double curve
Bumpy road
Slippery road
Road narrows on the right
Road work
Traffic signals
Pedestrians
Children crossing
Bicycles crossing
Beware of ice/snow
Wild animals crossing
End of all speed and passing limits
Turn right ahead
Turn left ahead
Ahead only
Go straight or right
Go straight or left
Keep right
Keep left
Roundabout mandatory
End of no passing
End of no passing by vehicles over 3.5 metric tons



We can clearly see that the distributions of images between the different datasets are roughly the same. What's interesting to see is that some images are more scarce than others, signifying that some signs rarely occur on the road as they would be edge cases that rarely do show up and intuitively this makes sense.

As a final means of exploration, the figure below shows 16 randomly sampled images from the training set.



We can see that the signs are contained within a variety of backgrounds, lighting conditions and can appear at any location in the image. A good classification algorithm should be robust against these.

Design and Test a Model Architecture

Preprocessing steps

I decided to leave the colour information in as they are powerful visual cues in how we identify signs. Red signs mean to proceed with caution while yellow or green signs are to be taken less seriously. Converting to grayscale would lose this semantic information. However, I did normalise the images so that the range of intensities spans from `[-1,1]`. Normalisation helps with training neural networks as it would be safer in terms of precision and training. If the raw activations of the layers were to get too large, the minimisation would become numerically unstable. Therefore for normaisation, you simply subtract all values over all channels by 128, then divide by 128.

Model Architecture

I decided to follow an architecture outlined by Andrej Karpathy in his blog: https://cs231n.github.io/convolutional-networks/#architectures. In particular, this architecture spoke to me:

```
INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC
```

The network is not too deep, but deep enough to extract meaningful features from images that are as small as 32 x 32. In addition, I introduced dropout right before the fully connected layers to mitigate any overfitting

issues. As I went deeper, I introduced more filters to the fold. Convolutional layers are well suited for this problem as we want to identify features spatially in an image that would better allow us to differentiate between the different signs. We want to find an optimal set of filters that would garner a high response on unique areas in the image that would optimally describe each sign. The fully-connected layers are standard to allow us to use the features we extracted from the convolutional layers to best produce the probabilities of identifying which sign we're looking at.

Therefore, my final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x16
RELU	
Max pooling	2x2 stride, outputs 14x14x16
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x32
RELU	
Max pooling	2x2 stride, outputs 5x5x32
Flatten	Outputs a 800 element tensor
Dropout	Keep probability of 0.5
Fully connected	512 output neurons
RELU	
Dropout	Keep probability of 0.5
Fully connected	43 output neurons corresponding to 43 classes
Softmax	43 output neurons describing the probability of each class

Model Training

To optimise this neural network, the Adam optimiser was used with a learning rate of 0.001. The batch size I used was 64, the number of epochs used was 50 and the loss function use was the *sparse* categorical cross-entropy function. The reason why is because this avoids unnecessarily transforming the class labels into one-hot encoded vectors. The sparse categorical cross-entropy loss function expects the integer ID of the ground truth target, so it makes computing the cross entropy more stable. The choice of these parameters was mostly due to trial and error, combined with the hardware resources I have access to (I am using a P100 GPU) making a balance between speed and computational efficiency. I set up a model checkpoint callback such that we save the model with the highest validation accuracy over the epochs. Even though we finish training at 50 epochs which may be considered much longer than most systems that train with the resolution of these kinds of images, there will inevitably have been an epoch before this point that achieved a higher validation accuracy than what was seen at the end so we will use whatever model is saved that has the highest validation accuracy.

Therefore, training with a large number of epochs, but enforcing a callback to save the model with the highest validation accuracy is a form of early stopping.

Solution Approach

Using the guidelines from the blog post above, my results I achieved with each set were:

- Training set accuracy of 0.9746
- Validation set accuracy of 0.9807
- Test set accuracy of 0.9621

The training and validation accuracy can be found in the cell where the training occurs, and the test set accuracy can be found after the model training and it finally being saved to disk in the notebook.

This performance was achieved at epoch 14 out of 50 in the training. Therefore, we exceed the validation accuracy requirement of 0.93 on the first try. I believed this model would work mainly because it was advocated by a well-known expert, but the design and architecture made the most sense to me given how small the image is and how expressive we can be with a small image. I am convinced that the system is performing as expected because the magnitude of the accuracy for the training, validation and test sets are relatively the same. This is indicative of the the system generalising and not overfitting.

Test a Model on New Images

I found six German traffic signs on the web but they were very large in resolution. I took it upon myself to crop the images then resized them to 32 x 32 then cropped them so that they would be suitable for the architecture I designed above.

Here are the six German traffic signs that I found on the web:

The collection of these may be difficult to classify as all of them have varying backgrounds. In addition, some of them have some slight orientation which we did not introduce in the training. In particular, we did not introduce any augmentation. I also decided to find a sign that was completely blank but only had a red outer border, which signifies that no vehicles are allowed. This is indeed in our dataset, but the highly ambiguous nature of the red and white circular borders may make it harder to classify because there are many signs that do have this, but are only differentiated with the text inside the region.

The top 5 predictions for each of the images is shown below. This code is run in the Output Top 5 Softmax Probabilities section in the notebook. Additionally, the inference is done in the Predict the Sign Type for Each Image section in the notebook.

```
Image #1

Class #1 - Speed limit (30km/h) - Probability: 1.0
Class #2 - Speed limit (50km/h) - Probability: 1.2542992200437197e-09
Class #3 - Roundabout mandatory - Probability: 4.2202660810453096e-10
```

```
Class #4 - Keep right - Probability: 1.1772216534922109e-10
Class #5 - End of speed limit (80km/h) - Probability: 7.537577333127654e-
12
```

```
Image #2

Class #1 - Bumpy road - Probability: 0.9999980926513672
Class #2 - Bicycles crossing - Probability: 1.9274636997579364e-06
Class #3 - No vehicles - Probability: 4.024740818397987e-10
Class #4 - Road work - Probability: 8.699659916067137e-12
Class #5 - Traffic signals - Probability: 6.647298163298121e-12
```

```
Image #3

Class #1 - Ahead only - Probability: 0.9999998807907104
Class #2 - Go straight or right - Probability: 1.2893966072624607e-07
Class #3 - Turn right ahead - Probability: 1.7854874523326458e-10
Class #4 - Road work - Probability: 2.92059362805297e-11
Class #5 - Roundabout mandatory - Probability: 1.2371377793723992e-12
```

```
Image #4

Class #1 - Turn right ahead - Probability: 0.4148518145084381
Class #2 - No vehicles - Probability: 0.39023715257644653
Class #3 - Roundabout mandatory - Probability: 0.14847058057785034
Class #4 - Keep right - Probability: 0.04245895519852638
Class #5 - Ahead only - Probability: 0.002139888470992446
```

```
Image #5

Class #1 - Go straight or left - Probability: 1.0
Class #2 - Roundabout mandatory - Probability: 2.3739414947709925e-11
Class #3 - Keep left - Probability: 3.119415668698458e-15
```

```
Class #4 - Turn right ahead - Probability: 8.128294921463757e-17
Class #5 - General caution - Probability: 3.731395613204763e-18
```

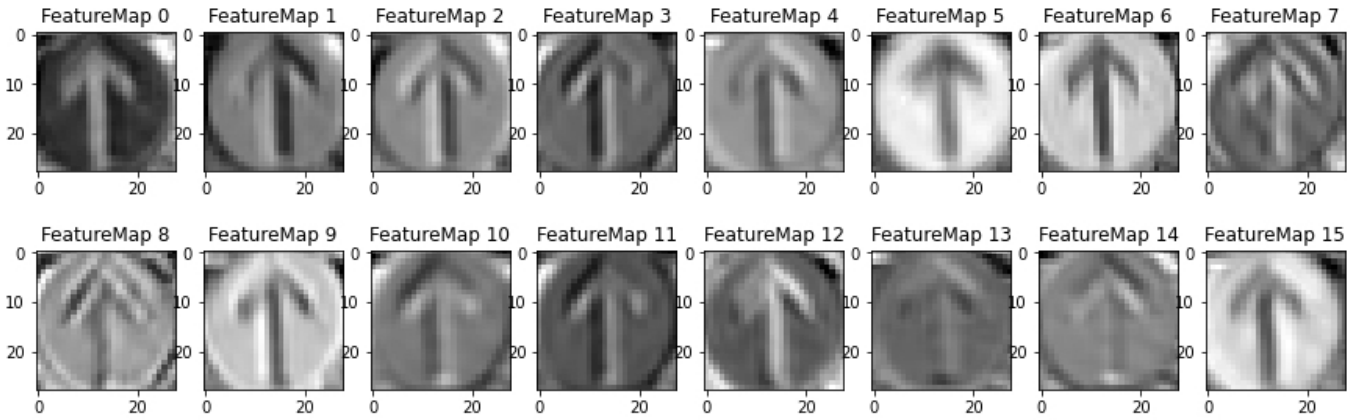
```
Image #6

Class #1 - General caution - Probability: 1.0
Class #2 - Pedestrians - Probability: 1.2275169709380838e-24
Class #3 - Traffic signals - Probability: 1.2176799961729185e-33
Class #4 - Go straight or left - Probability: 1.5617133512782678e-34
Class #5 - Roundabout mandatory - Probability: 1.0926069861380888e-36
```

For all of the images shown above, most of them were extremely confident for the most probable class having probabilities of close to 1.0. The only image that showed some confusion was the fourth one where no vehicles were allowed. Turn right and no vehicles were close in their probabilities. For the rest of the results, the other probabilities were close to 0, so our model was able to generalise well given this new data. As such, our accuracy for this small dataset is 5 / 6 or 0.8333... The performance of 5 / 6 compared to the test dataset of ~0.96 is lower in performance, but with a larger sampling of images found on the Internet I am confident the performance can go higher in comparison to the test dataset. So in summary:

Image	Prediction
30 km/h	30 km/h
Bumpy Road	Bumpy Road
Ahead Only	Ahead Only
No vehicles	Turn right ahead
Go straight or left	Go straight or left
General caution	General caution

Finally, here's a sample of what the activations looked like after the first convolutional layer for the ahead only sign.



We can interpret each feature map such that dark pixels indicate low scores or what the filter is not interested in whereas light pixels indicate high scores or what the filter is very interested in. We can see that there is a mix of filters that very much are interested in this sign and filters that aren't. This makes sense as there are signs that look visually similar to the look ahead sign and we need more stronger visual cues down the line to help us discern between the signs.