Traffic Sign Recognition

Writeup

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

Build a Traffic Sign Recognition Project

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

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

You're reading it! and here is a link to my project code.

Data Set Summary & Exploration

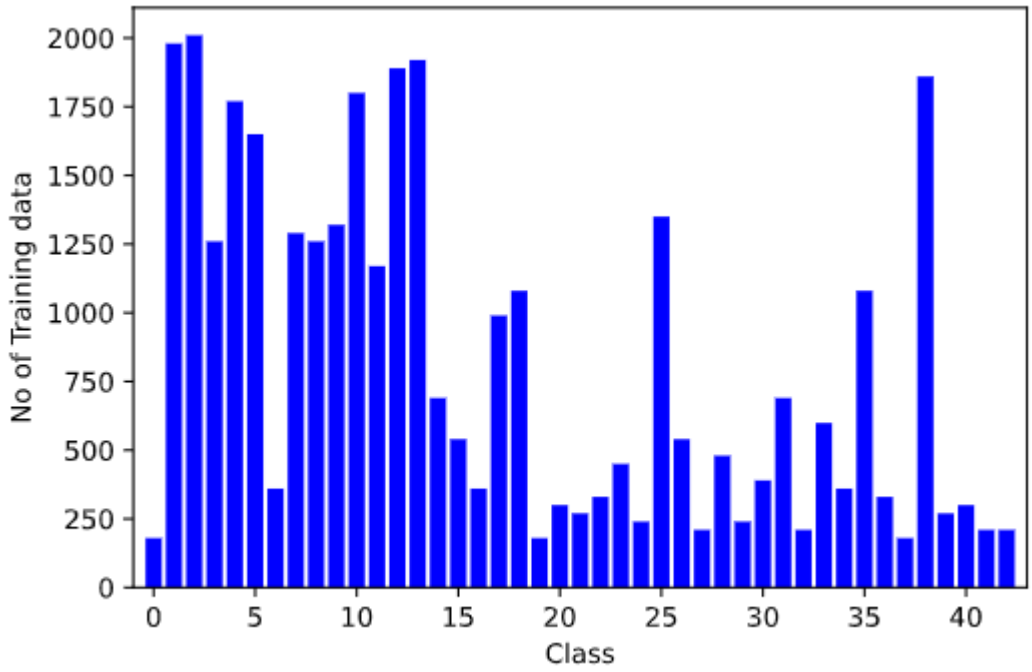
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

The pandas library was used to calculate summary statistics of the traffic signs data set:

- 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, 32, 3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

A histogram of the training set by classification label was created. This revealed that the training set is unbalanced in regards to the number of examples provided for each of the classification labels.



Bellow some an images of each class is shown to get an idea of what they look like in the given format.

Design and Test a Model Architecture

1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

1. Dataset Preprocessing

1.1 Data Augmentation

From the exploration of the data can be seen that the classes as unbalanced. Therefore it was decided to augmentate the training data set to balance the clases to have a minimum of 2000 images per clases.

The augmentation procedure it's done by the method `generate_Image(images,images_to_generate)` based on this project.

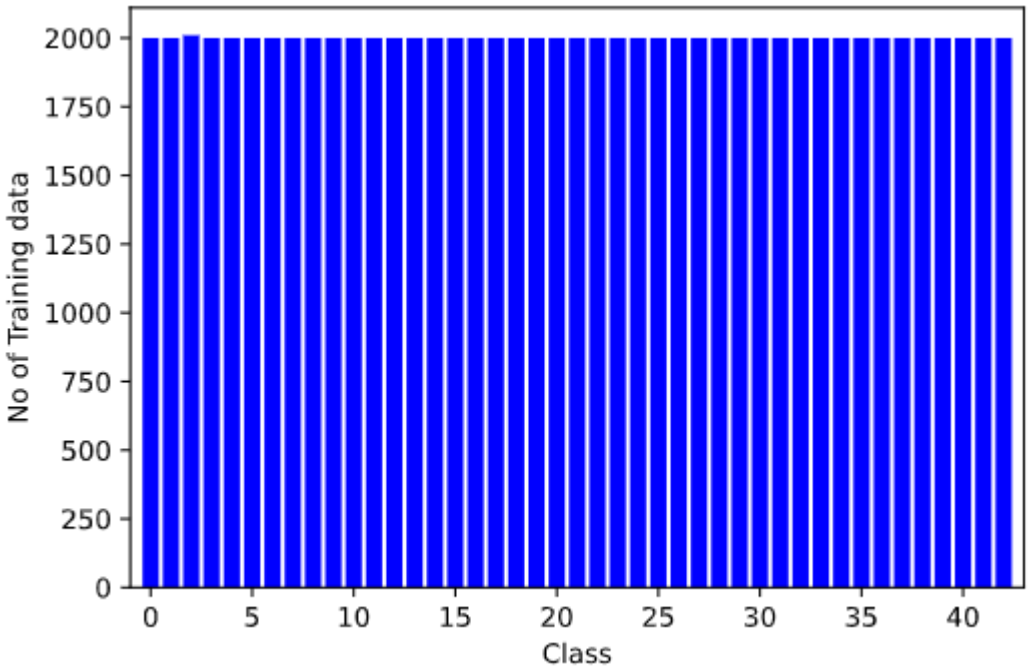
Up to four different transformations can be applied to an images:

1. Random rotation between -20 deg and 20 deg.
2. Random noise.
3. Gaussian Blur with a kernel of 3.
4. Warp Shifting of a random distance from -5 to 5 pixels in the X and Y direction.

The method `generate_Image(images,images_to_generate)` recieves as input the images of the Class to be augmented and the number of images to be generated, and the functionality is as follows:

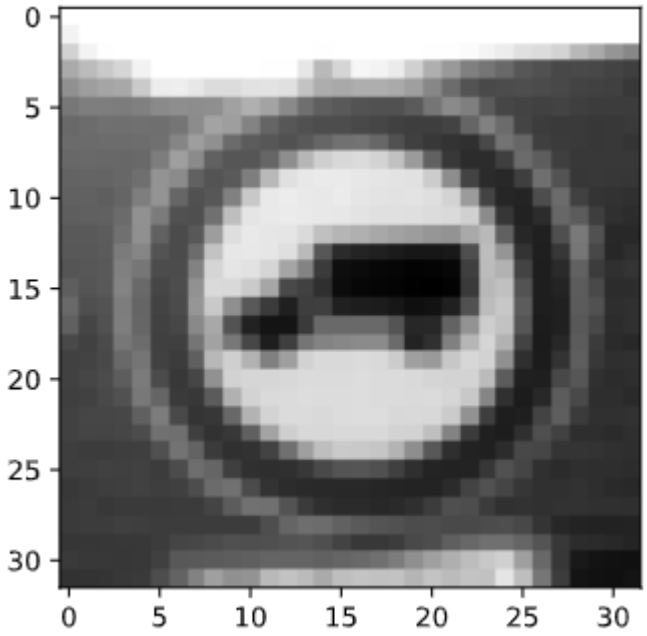
1. The method iterates form `0` to `images_to_generate`.
2. On each iteration a random images from the set is chosen to be transformed.
3. A random number of transformation to be apply to the image is chosen. (between `0` and `4`, [ammount of different transformation]).
4. The transformations to be made are randomly choosen.

After Augmentation the class Histograms looks as follows:

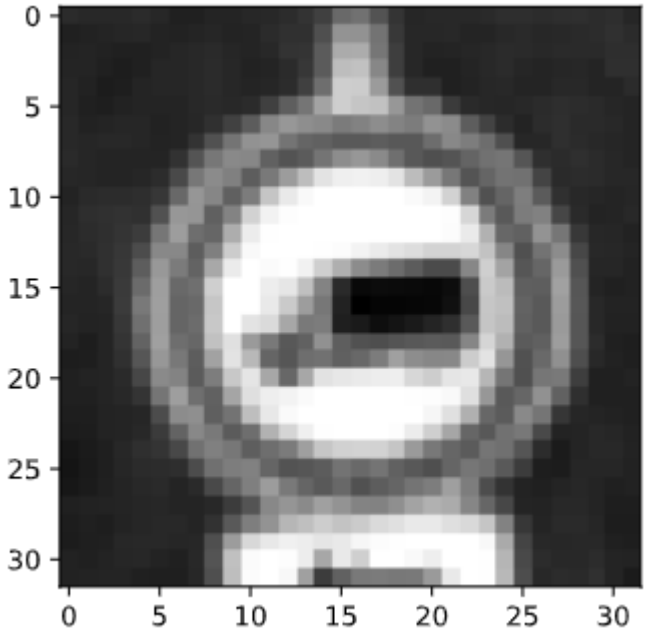


1.2 Pre-processing

One obvious preprocessing step is to convert the color images into a more compact format that retains the visual features of the images. Grayscale is a good candidate, and it is the conversion which was used in this project. Below is an example image after grayscale conversion:



After transforming the images to gray scale, these are statarized to have Zero mean and standard deviation of 1 using `pixel = (pixel - mean) / std`. Below is an example image after grayscale conversion and standarization:



2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grayscale Image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU Activation	

Layer	Description
Max Pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
RELU Activation	
Max Pooling	2x2 stride, outputs 5x5x16
Flatten & Concatenate Conv1 and Conv2	Output a single vector of size 1576
Fully Connected	Outputs 150
RELU Activation	
Fully Connected	Outputs 90
RELU Activation	
Fully Connected	Outputs 43

The LeNet architecture was adapted for this problem. A few suggestions where taken from this paper. Particularly the suggestion to concatenate the outputs of each of the two convolution layers and feed them both into the fully connected layers prior to the classification logits.

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

To train the model, the softmax cross entropy was used as the cost function, and an AdamOptimizer. This minimized the mean of cross entropy over 40 epochs of training, with a batch size of 128 examples. A variable learning rate was used, starting with a rate of `0.0015` and reducing by `10%` every 10 epochs. for the first convolution layer a dropout of `10%` was applied, for the second convolution layer a `25%`, and a dropout of `50%` was used for the fully connected layes. These dropuot rates where chosen following the recommendation ofthis forum.

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

My final model results were:

- validation set accuracy of 96.6%
- test set accuracy of 93.4%

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?

The LeNet architecture was chosen because it was familiar to me and was designed to solve a similar classification problem using image data as input.

- What were some problems with the initial architecture?

At first the architecture flattening just the second convolution was used, but it was not capable or reaching the minimum requirements of the project, also it did not distiguish well between features of small size for example between such as the bicycle-crossing and children-crossing signs.

- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

One major improvement that was made was to concatenate the outputs of both of the convolution layers and submit them as input into the fully-connected layers. also the application of the different dropout rates for the different layers. This improved my overall performance, as well as improving the training time necessary to reach convergence upon a relative maxima in accuracy.

- Which parameters were tuned? How were they adjusted and why?

- Tuned some of the layer sizes in the fully-connected layers to accomodate for more information coming from the concatenated convolution outputs.
- The progressing decay of the learning rate.
- the tunning of the dropout parameter to avoid overfitting.

- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

A convolution layer is useful for distinguishing the features of each class of traffic sign regardless of their absolute position within the input. The resolution of each convolution allowed the model to discover features at their respective input scale, with the first convolution discovering quite small features and the second convolution discovering larger features.

If a well known architecture was chosen:

- What architecture was chosen?

the LeNet architecture was chosen and then adapted to the needs of this project's application.

- Why did you believe it would be relevant to the traffic sign application?

Even though the LeNet architecture is mostly used for handwriting classification problems, which relies more on distinction of a single set of important features (pen strokes), I believed that shuch architcture would be of use to solve the traffic sign classification because it involved image recognition/classification. In adapting the LeNet model architecture to the traffic sign classification problem, careful changes where made to allow the classifier to act upon multiple groups of features. This adaptation was suggested by the provided paper authored by Pierre Sermanet and Yann LeCun

- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

The model's accuracy on all three sets indicates that the model performs well on examples to which it has been exposed as well as generalizing well to examples which it has not experienced before in training. The test set accuracy in particular gave good confidence that this model generalizes well to new examples in the same format.

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are sixteen German traffic signs that where found on the web:

The images where chosen in groups, in order to be able to test how well th model worked. by choosing images of different speed limits the model needs to be able to diferentiate smaller features, the same happens for the two images of overtaking.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

Here are the results of the prediction:

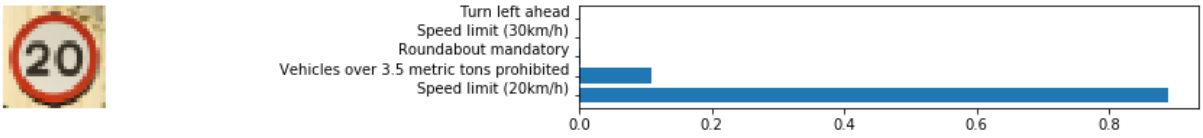
The model was able to correctly guess 15 out of the 16 traffic signs, which gives an accuracy of 93.8%. This compares favorably to the accuracy on the test set of 93.4%

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

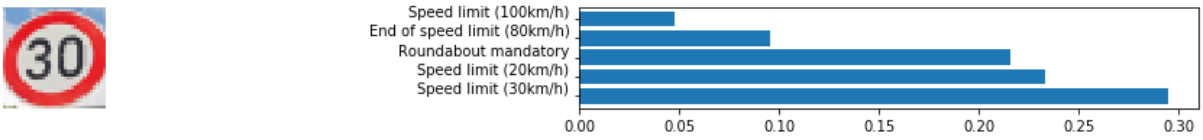
The code for making predictions on my final model is located in the 22th cell of the Ipython notebook.

The top 5 probabilities of each images are shown below.

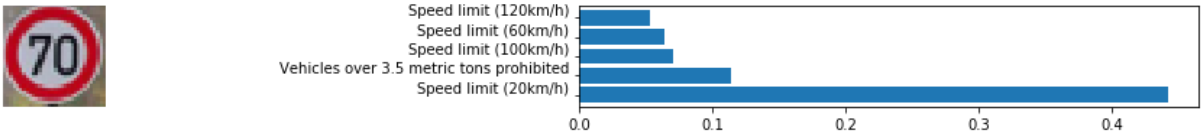
Speed limit (20km/h)



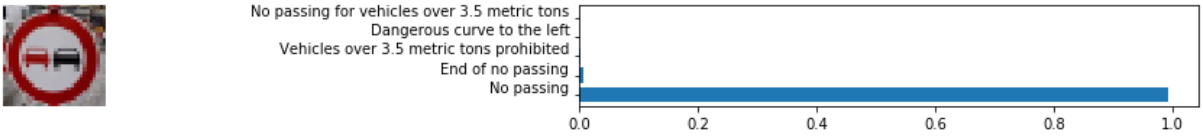
Speed limit (30km/h)



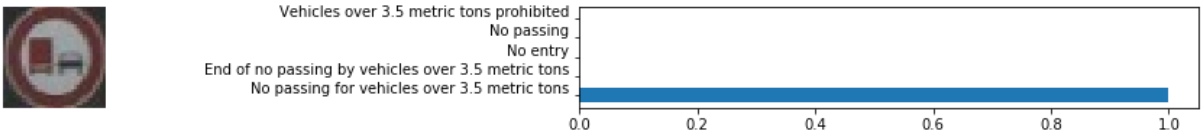
Speed limit (20km/h)



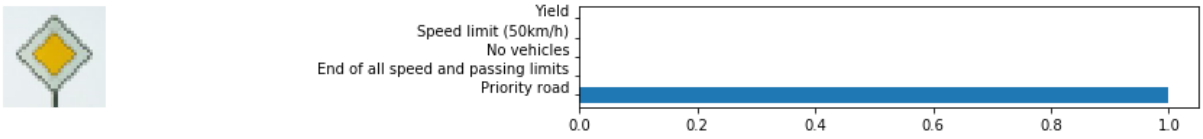
No passing



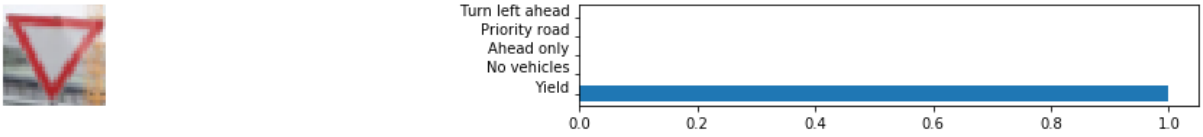
No passing for vehicles over 3.5 metric tons



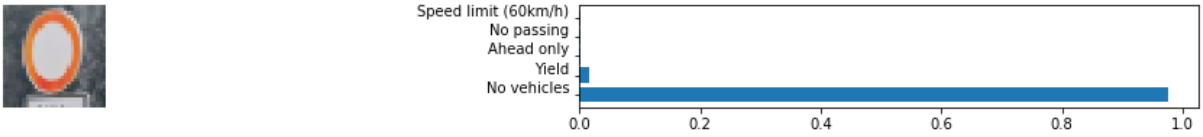
Priority road



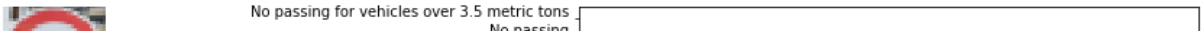
Yield

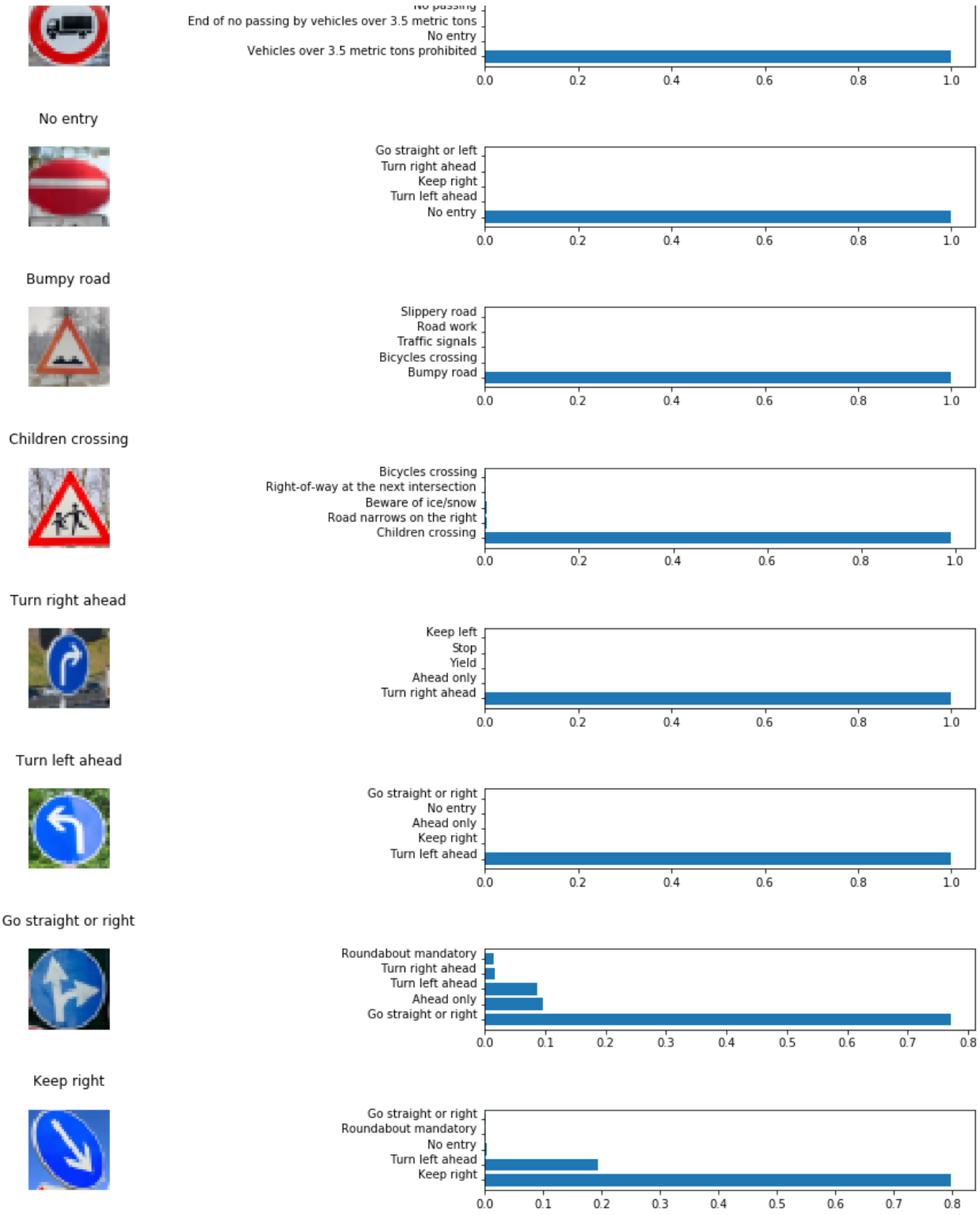


No vehicles



Vehicles over 3.5 metric tons prohibited





(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?