Traffic Sign Recognition

By Brian Meier Udacity Self-Driving ND 3/6/172

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.

0. 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!

Data Set Summary & Exploration

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

The code for this step is contained in the code cells under the 'Provide a Basic Summary of the Data Set Using Python, Numpy and/or Pandas' heading of the iPython notebook.

I used the numpy library to calculate summary statistics of the traffic

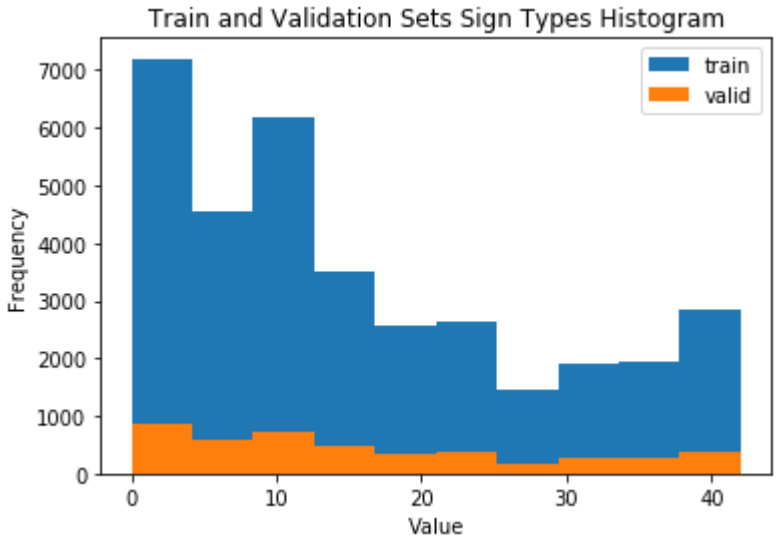
signs data set:

- The size of training set is 34,799
- The size of validation set is 4,410
- The size of the test set is 34,799
- 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 and identify where the code is in your code file.

The code for this step is contained under the ‘Include an exploratory visualization of the data set’ cell in the IPython notebook.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed for both the training and validation sets.



Design and Test a Model Architecture

1. Describe how, and identify where in your code, 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.

The code for this step is also contained under the 'Include an exploratory visualization of the data set' cell in the IPython notebook.

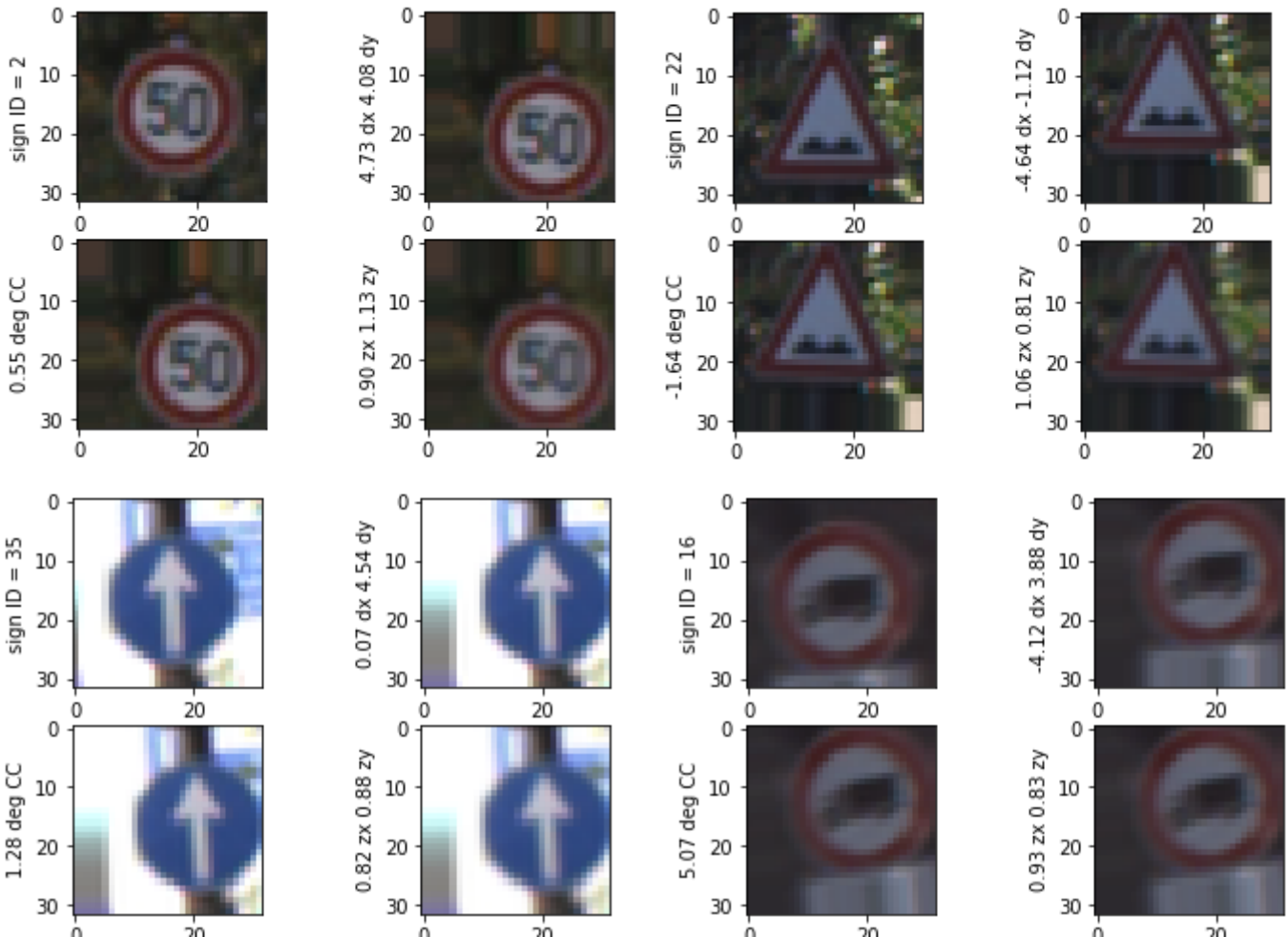
I first shifted the image between +- 5 pixels because signs can be a different locations in an image.

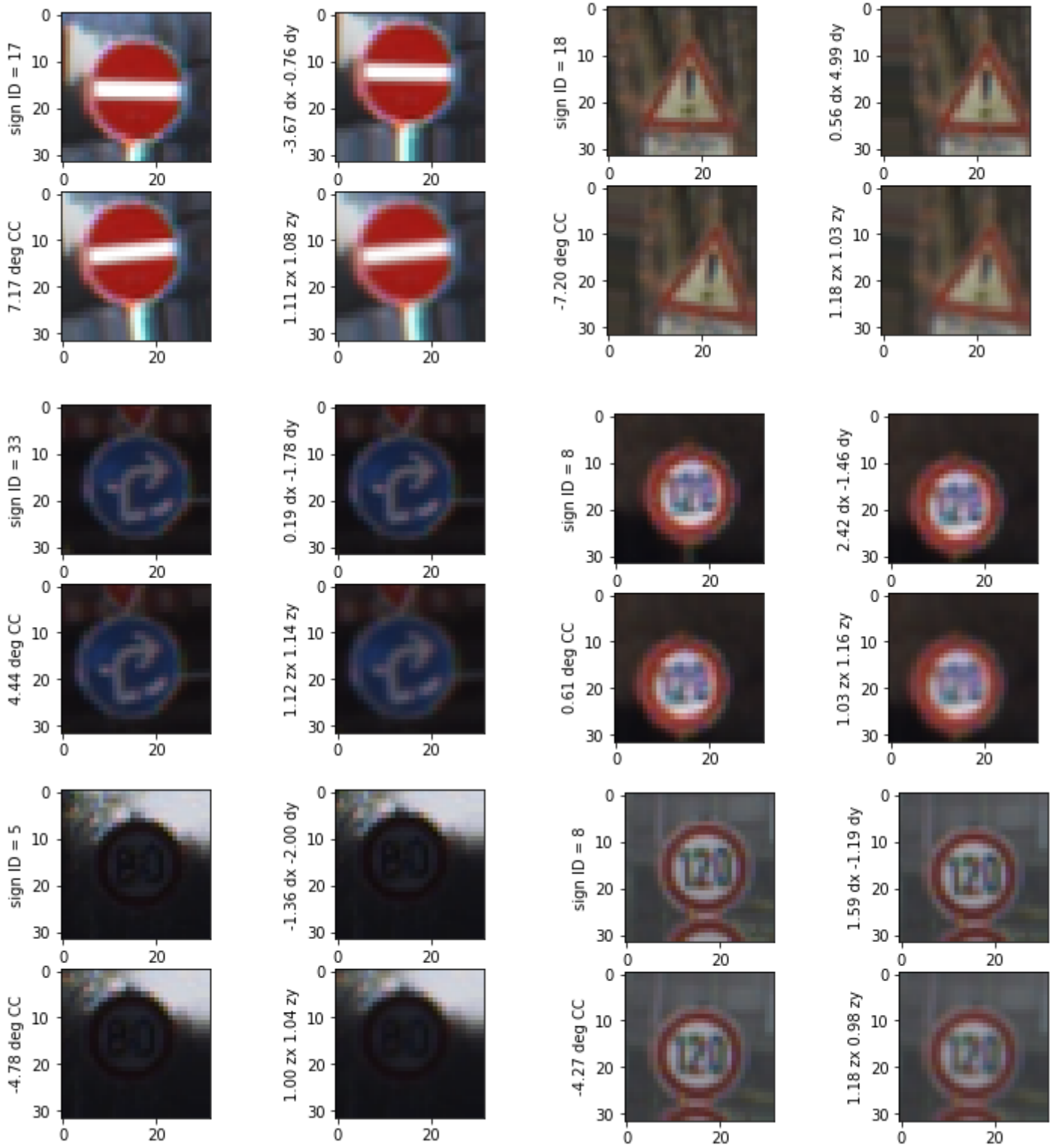
Next, I rotated the image between +- 10 degrees because images can be taken from different rotations, and signs can rotate.

Then, I distorted the rotated image by zooming in the x and y directions by factors ranging between 0.8 to 0.2 because as on passes a sign, it can become squished.

Finally, I resized the sign to a 32 x 32 size so it can be fed into the model.

Below are results of these transformations on 10 signs.




sign ID = 17
-3.67 dx -0.76 dy
sign ID = 18
0.56 dx 4.99 dy
7.17 deg CC
1.11 zx 1.08 zy
-7.20 deg CC
1.18 zx 1.03 zy
sign ID = 33
0.19 dx -1.78 dy
sign ID = 8
2.42 dx -1.46 dy
4.44 deg CC
1.12 zx 1.14 zy
0.61 deg CC
1.03 zx 1.16 zy
sign ID = 5
-1.36 dx -2.00 dy
sign ID = 8
1.59 dx -1.19 dy
-4.78 deg CC
1.00 zx 1.04 zy
-4.27 deg CC
1.18 zx 0.98 zy

To avoid streaming, I choose to keep the files small by storing the values as uint8 and not normalizing outside of the graph.

2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

The pickled files were separated into training, validation and test sets. I did not change them.

Examples of original and augmented images are given above. The training set was augmented 11 times to result in a total of 417,588 training images. These were generated in the cells under the 'Generate and save additional data' cell in the ipython notebook.

3. Describe, and identify where in your code, 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.

The code for my final model is located under the 'Create Graph' cell of the Ipython notebook.

My final model consisted of the following layers contains the layers given in the table below. It was inspired by the VGGNet architure.

RELU operations follow all CONV and the FC1 and FC2 layers.

Layer	H	W	D	Memory	Weights
Input	32	32	3	3,072	0
CONV 3x3	32	32	16	16,384	432
CONV 3x3	32	32	16	16,384	2,304
CONV 3x3	32	32	16	16,384	2,304
CONV 3x3	32	32	16	16,384	2,304
MAX POOL 2x2	16	16	16	4,096	0
CONV 3x3	16	16	32	8,192	4,608
CONV 3x3	16	16	32	8,192	9,216
CONV 3x3	16	16	32	8,192	9,216
CONV 3x3	16	16	32	8,192	9,216
MAX POOL 2x2	16	16	32	8,192	0
CONV 3x3	8	8	64	4,096	18,432
CONV 3x3	8	8	64	4,096	36,864
CONV 3x3	8	8	64	4,096	36,864
CONV 3x3	8	8	64	4,096	36,864
FC 1 w dropout	1	1	1024	1,024	4,194,304
FC 2 w dropout	1	1	1024	1,024	1,048,576
FC output	1	1	43	43	44,032

4. Describe how, and identify where in your code, 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.

The code for training the model is located in the next cell of the Ipython notebook.

To train the model, I used:

Item	Value
EPOCHS	3
BATCH_SIZE	32
Optimizer	Adam, rate = 0.0005
Dropout	0.25
L2 Loss for FC layers	Beta = 0.01
Loss	Softmax of cross entropy with logist plus an L2 loss for FC weights

5. Describe the approach taken for finding a solution. 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.

The code for calculating the accuracy of the model is located in the cell under the 'Test Results:' cell of the Ipython notebook.

My final model results were:

- training set accuracy of 0.998
- validation set accuracy of 0.987
- test set accuracy of 0.968

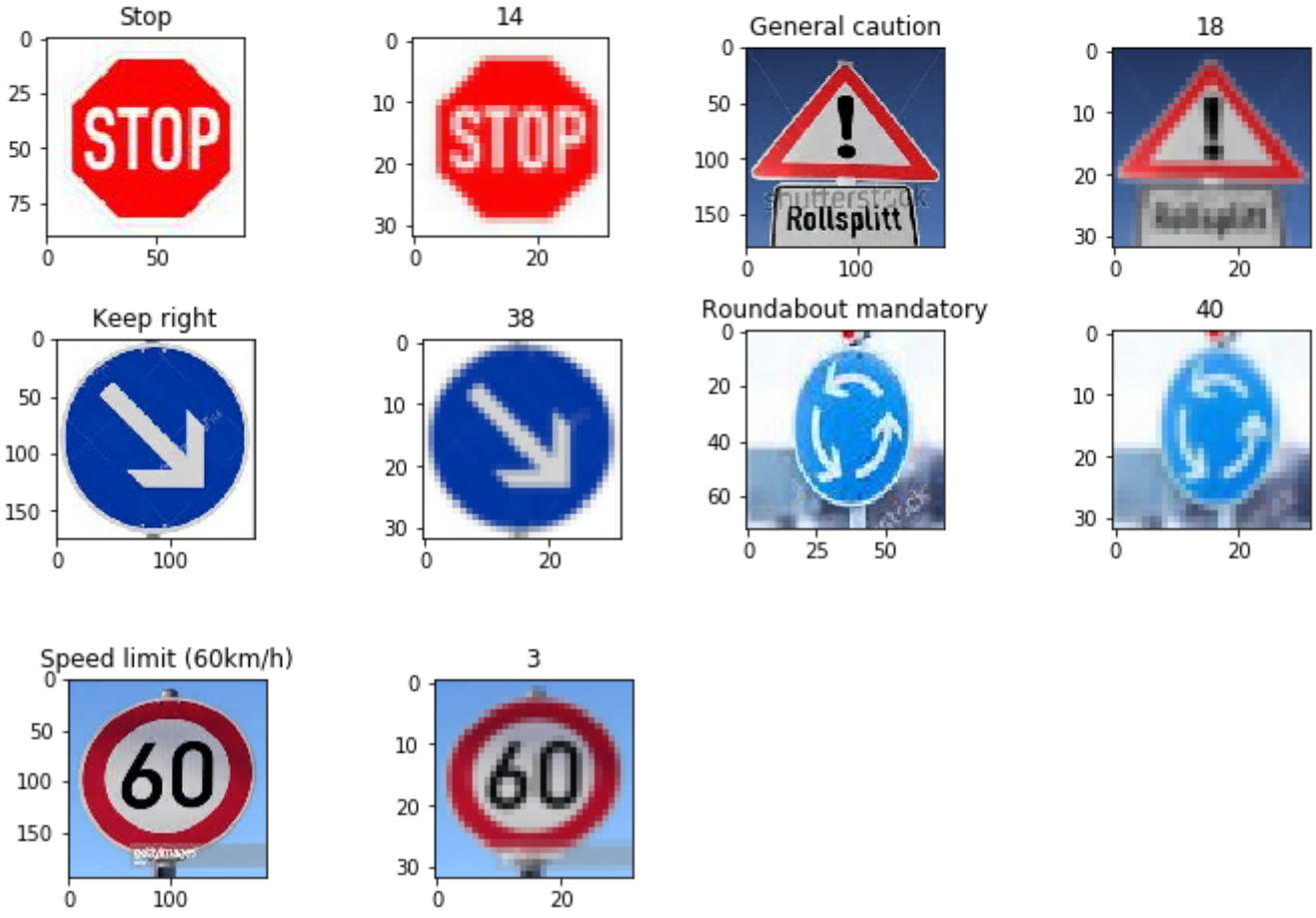
If a well known architecture was chosen:

- A variation of the VGGNet architecture was chosen.
- I believed it would do well on the traffic sign application because it finished second in the ILSVRC 2014 competition.
- The models performance on the training, validation and test set are all well above the 93% threshold set forth in the assignment. In addition to the final run, a total of seven trials were performed. Of these, they contained 10 EPOCHS twice, 5 EPOCHS three times and 3 EPOCHS twice. From EPOCHS 3+, the average validation accuracy is 0.984 and the standard deviation is 0.0043. The average test accuracy for the seven trials is 0.969 and the standard deviation is 0.0073.

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.

Five images found on the web are listed below. I expected all of them to be easy to classify.

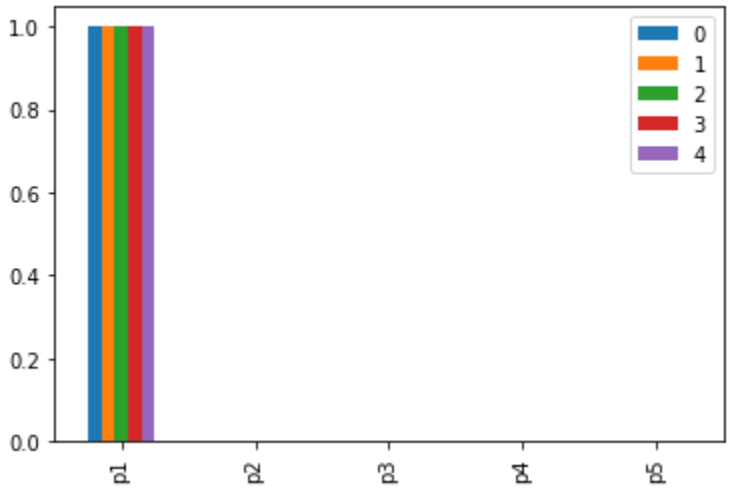


2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the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

The code for making predictions on my final model is located in the cell below the ‘Sign predictions, accuracy, and top five selections and probabilities for web images’ cell of the Ipython notebook.

The test accuracy was 100%.

Index	14	18	38	40	3
PredInd1	14	18	38	40	3
PredInd2	15	27	39	23	10
PredInd3	27	39	24	19	33
PredInd4	17	16	6	17	16
PredInd5	23	23	5	15	20
p1	1	1	1	1	1
p2	5.80E-11	3.11E-25	1.45E-27	4.16E-18	3.37E-22
p3	4.42E-11	5.88E-26	6.36E-28	8.88E-19	6.12E-23
p4	3.76E-11	5.55E-26	1.43E-28	8.00E-19	2.36E-23
p5	3.68E-11	3.56E-26	8.20E-29	5.17E-19	1.64E-23



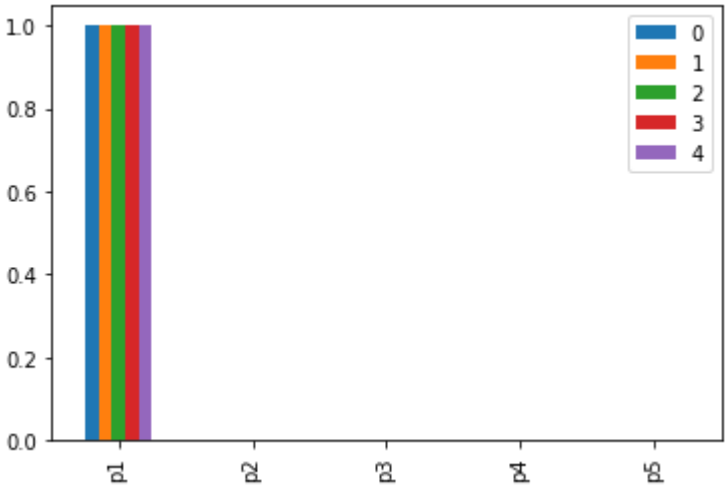
Because each image was clear, the model was extremely confident.

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 and identify where in your code softmax probabilities were outputted. 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)

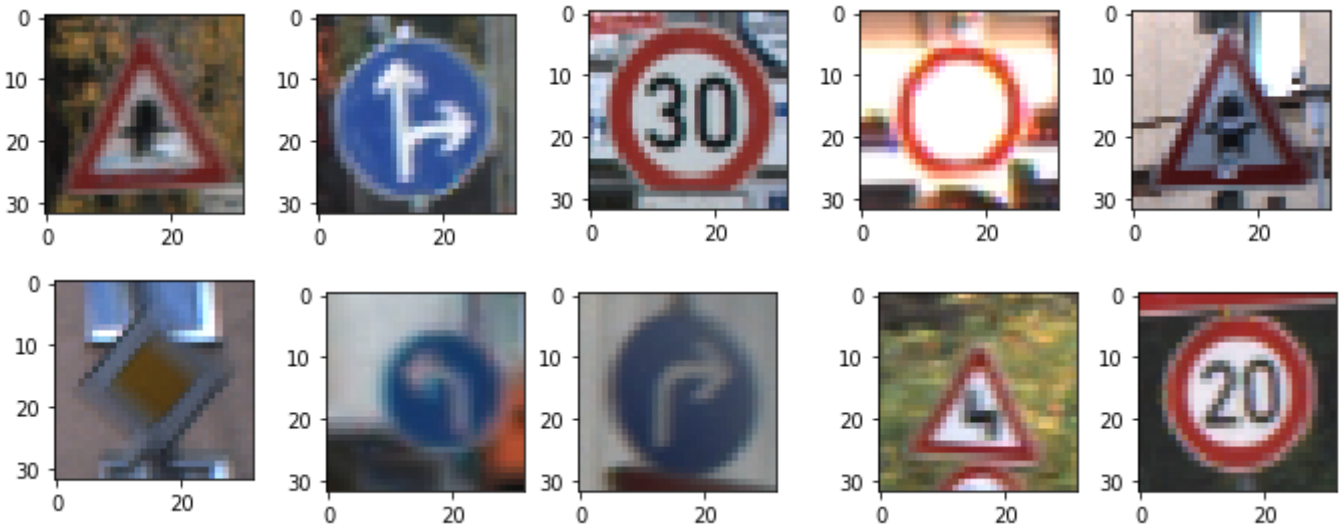
The code for making predictions on my final model is located in the cell below the ‘Sign predictions, accuracy, and top five selections and probabilities for web images’ cell of the Ipython notebook.

The test accuracy was 100%.

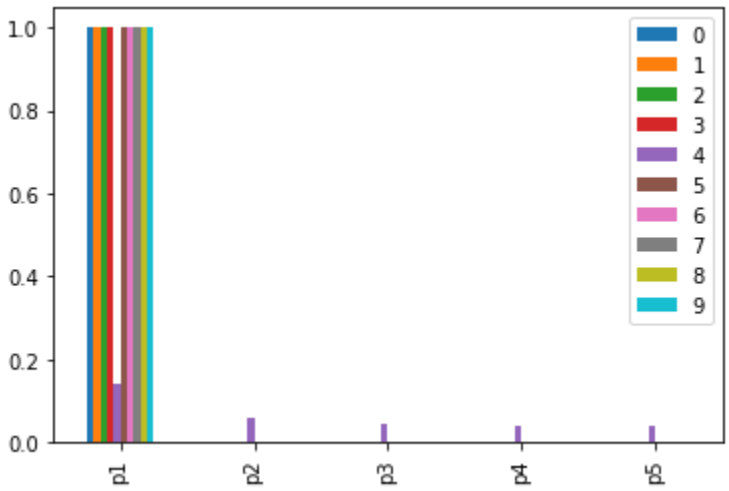
Index	14	18	38	40	3
PredInd1	14	18	38	40	3
PredInd2	15	27	39	23	10
PredInd3	27	39	24	19	33
PredInd4	17	16	6	17	16
PredInd5	23	23	5	15	20
p1	1	1	1	1	1
p2	5.80E-11	3.11E-25	1.45E-27	4.16E-18	3.37E-22
p3	4.42E-11	5.88E-26	6.36E-28	8.88E-19	6.12E-23
p4	3.76E-11	5.55E-26	1.43E-28	8.00E-19	2.36E-23
p5	3.68E-11	3.56E-26	8.20E-29	5.17E-19	1.64E-23



This effort was repeated for 10 test images. The images are given below.



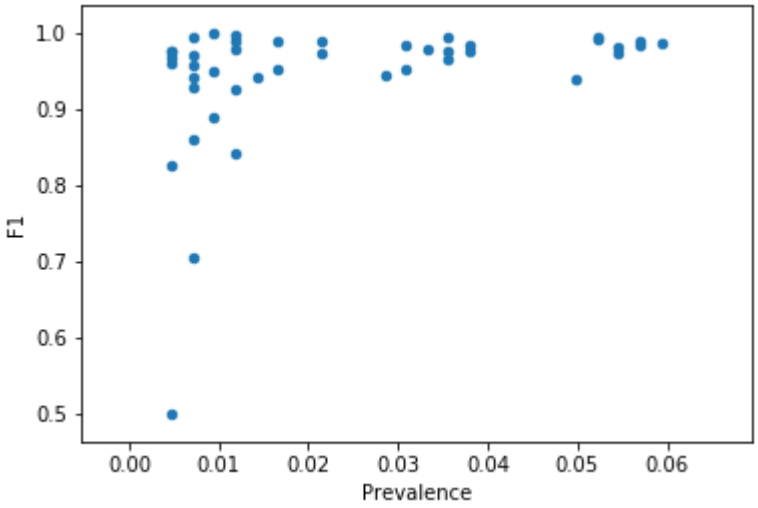
Index	11	36	1	15	11	12	34	33	21	0
PredInd1	11	36	1	15	27	12	34	33	21	0
PredInd2	39	0	2	6	1	34	12	39	31	24
PredInd3	41	1	23	23	5	0	21	3	24	33
PredInd4	31	2	6	39	21	1	22	36	34	23
PredInd5	19	3	10	41	24	2	17	24	23	36
p1	1	1	1	1	0.142	1	1	1	1	1
p2	4.46E-31	0.00E+00	7.39E-24	2.10E-44	0.058	5.38E-43	3.18E-22	1.30E-29	2.22E-17	1.02E-15
p3	3.06E-31	0.00E+00	3.80E-25	1.26E-44	0.045	0.00E+00	8.92E-24	1.14E-29	1.19E-17	4.12E-16
p4	4.88E-32	0.00E+00	2.06E-25	1.40E-45	0.039	0.00E+00	6.91E-24	4.50E-30	9.20E-18	3.99E-16
p5	9.99E-33	0.00E+00	4.93E-26	1.40E-45	0.038	0.00E+00	3.59E-24	8.20E-31	5.49E-18	3.82E-16



The test accuracy was 90%. It thought a Right-of-way at the next intersection was a Pedestrians sign. These are similar and the model was unsure which was correct.

Summary of Results

ClassId	SignName	Cond. Pos.	Pred. Pos.	True Pos.	Precision	Recall	F1	Prevalence
0	Speed limit (20km/h)	60	64	60	0.938	1.000	0.968	0.005
1	Speed limit (30km/h)	720	737	718	0.974	0.997	0.986	0.057
2	Speed limit (50km/h)	750	767	749	0.977	0.999	0.987	0.059
3	Speed limit (60km/h)	450	464	441	0.950	0.980	0.965	0.036
4	Speed limit (70km/h)	660	652	652	1.000	0.988	0.994	0.052
5	Speed limit (80km/h)	630	680	615	0.904	0.976	0.939	0.050
6	End of speed limit (80km/h)	150	133	131	0.985	0.873	0.926	0.012
7	Speed limit (100km/h)	450	447	446	0.998	0.991	0.994	0.036
8	Speed limit (120km/h)	450	434	432	0.995	0.960	0.977	0.036
9	No passing	480	494	479	0.970	0.998	0.984	0.038
10	No passing for vehicles over 3.5 metric tons	660	659	655	0.994	0.992	0.993	0.052
11	Right-of-way at the next intersection	420	406	404	0.995	0.962	0.978	0.033
12	Priority road	690	658	657	0.998	0.952	0.975	0.055
13	Yield	720	730	717	0.982	0.996	0.989	0.057
14	Stop	270	278	267	0.960	0.989	0.974	0.021
15	No vehicles	210	226	208	0.920	0.990	0.954	0.017
16	Vehicles over 3.5 metric tons prohibited	150	151	150	0.993	1.000	0.997	0.012
17	No entry	360	325	324	0.997	0.900	0.946	0.029
18	General caution	390	362	358	0.989	0.918	0.952	0.031
19	Dangerous curve to the left	60	63	60	0.952	1.000	0.976	0.005
20	Dangerous curve to the right	90	98	90	0.918	1.000	0.957	0.007
21	Double curve	90	80	60	0.750	0.667	0.706	0.007
22	Bumpy road	120	100	98	0.980	0.817	0.891	0.010
23	Slippery road	150	144	144	1.000	0.960	0.980	0.012
24	Road narrows on the right	90	97	87	0.897	0.967	0.930	0.007
25	Road work	480	469	464	0.989	0.967	0.978	0.038
26	Traffic signals	180	196	177	0.903	0.983	0.941	0.014
27	Pedestrians	60	64	31	0.484	0.517	0.500	0.005
28	Children crossing	150	153	150	0.980	1.000	0.990	0.012
29	Bicycles crossing	90	91	90	0.989	1.000	0.994	0.007
30	Beware of ice/snow	150	156	129	0.827	0.860	0.843	0.012
31	Wild animals crossing	270	275	270	0.982	1.000	0.991	0.021
32	End of all speed and passing limits	60	65	60	0.923	1.000	0.960	0.005
33	Turn right ahead	210	212	209	0.986	0.995	0.991	0.017
34	Turn left ahead	120	120	120	1.000	1.000	1.000	0.010
35	Ahead only	390	388	383	0.987	0.982	0.985	0.031
36	Go straight or right	120	128	118	0.922	0.983	0.952	0.010
37	Go straight or left	60	63	60	0.952	1.000	0.976	0.005
38	Keep right	690	664	664	1.000	0.962	0.981	0.055
39	Keep left	90	87	86	0.989	0.956	0.972	0.007
40	Roundabout mandatory	90	119	90	0.756	1.000	0.861	0.007
41	End of no passing	60	49	45	0.918	0.750	0.826	0.005
42	End of no passing by vehicles over 3.5 metric tons	90	82	81	0.988	0.900	0.942	0.007



Discussion of test results

From the above summary data and F1 vs. Prevalence plot, signs that are more highly represented tended to be better predicted. Also, the two worst performing signs -- Pedestrians and Double curve -- are both similar and have very low prevalence.

Conclusion

A CNN has been created that can detect German Street Signs.