Traffic Sign Recognition Project

Build a Traffic Sign Recognition Project

The goals of this project are the following:

- Design, train and test a model architecture for traffic sign classification
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

These are the steps taken to accomplish this project:

- Data set exploration and analysis
- Definition of a image preprocessing pipeline for image augmentation
- Generation of new images from the original **training** data set
- Network architecture definition
- Network training and evaluation based on training/validation dataset
- Network testing using the testing dataset
- Model evaluation with new images from the internet

Rubric Points

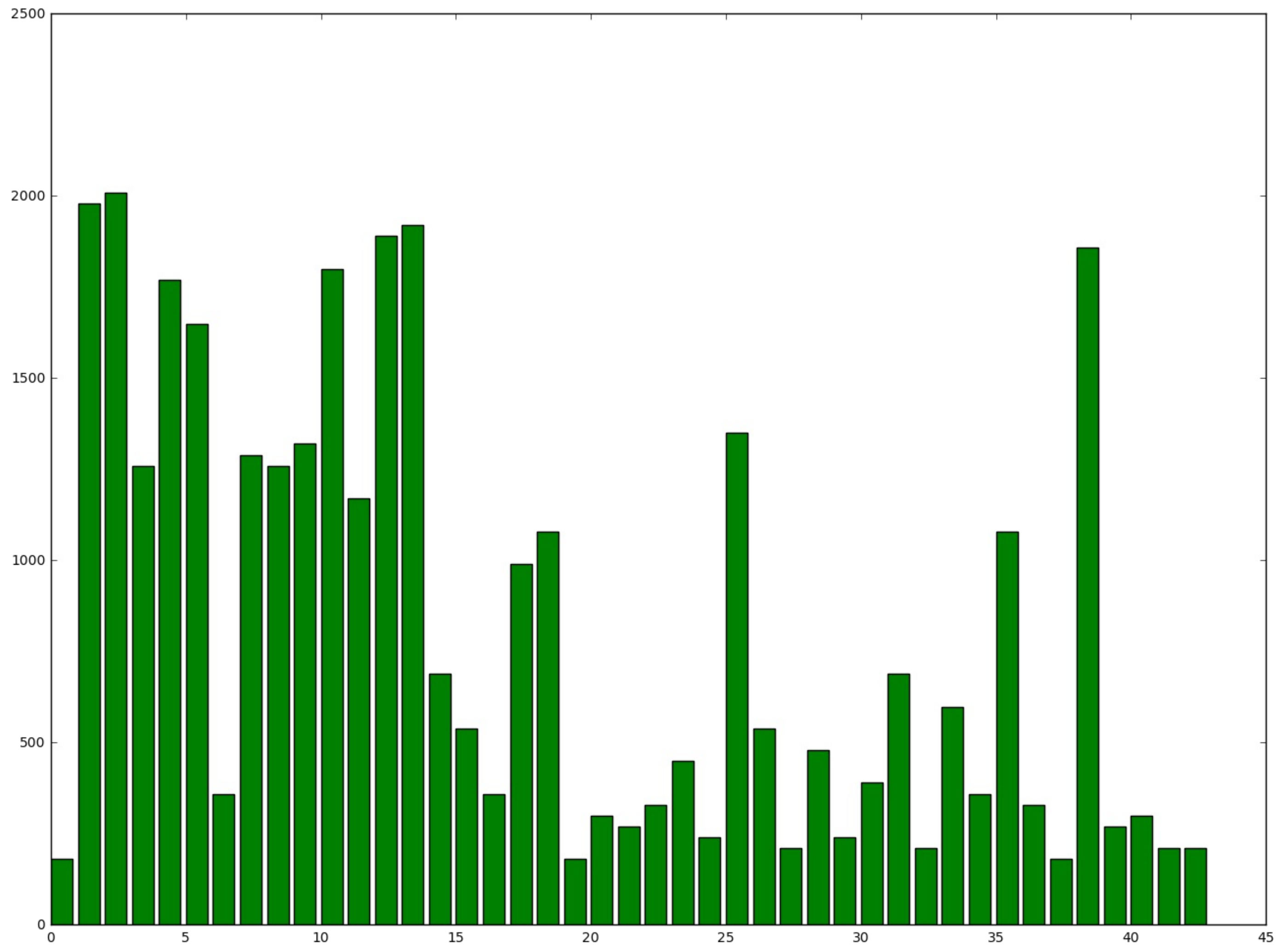
The following link will redirect you to my project notebook

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.

Cells #2 & #3 address this rubric requirement. The training data set has been evaluated and it has been observed that there were some classes which had as low as 10% of data if compared with the classes which had more data. This observation led the developer to generate new images in order to avoid data asymmetry issues. The image below presents the data distribution:



As one may observe, there is a large discrepancy regarding the amount of training samples each class has. In order to address this problem, new images are generated as an attempt to mitigate this issue. Below are some key values regarding the data set:

- The size of training set is 34799 images
- The size of test set is 12630 images
- The shape of a traffic sign image is [32, 32, 3]
- This problem has 43 distinct classes

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 in the cell #3 of the IPython notebook. The previous item already showed the training data distribution amongst the distinct classes and the image below contains one random sample of each class to illustrate the dataset. It is possible to observe that the images come in different sorts, shaded, bright, cut, different background and so on.

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 cell #4 of the IPython notebook contains some auxiliary function used on the data preprocessing pipeline, they are:

- rotateImage: rotates the image by a random degree between -

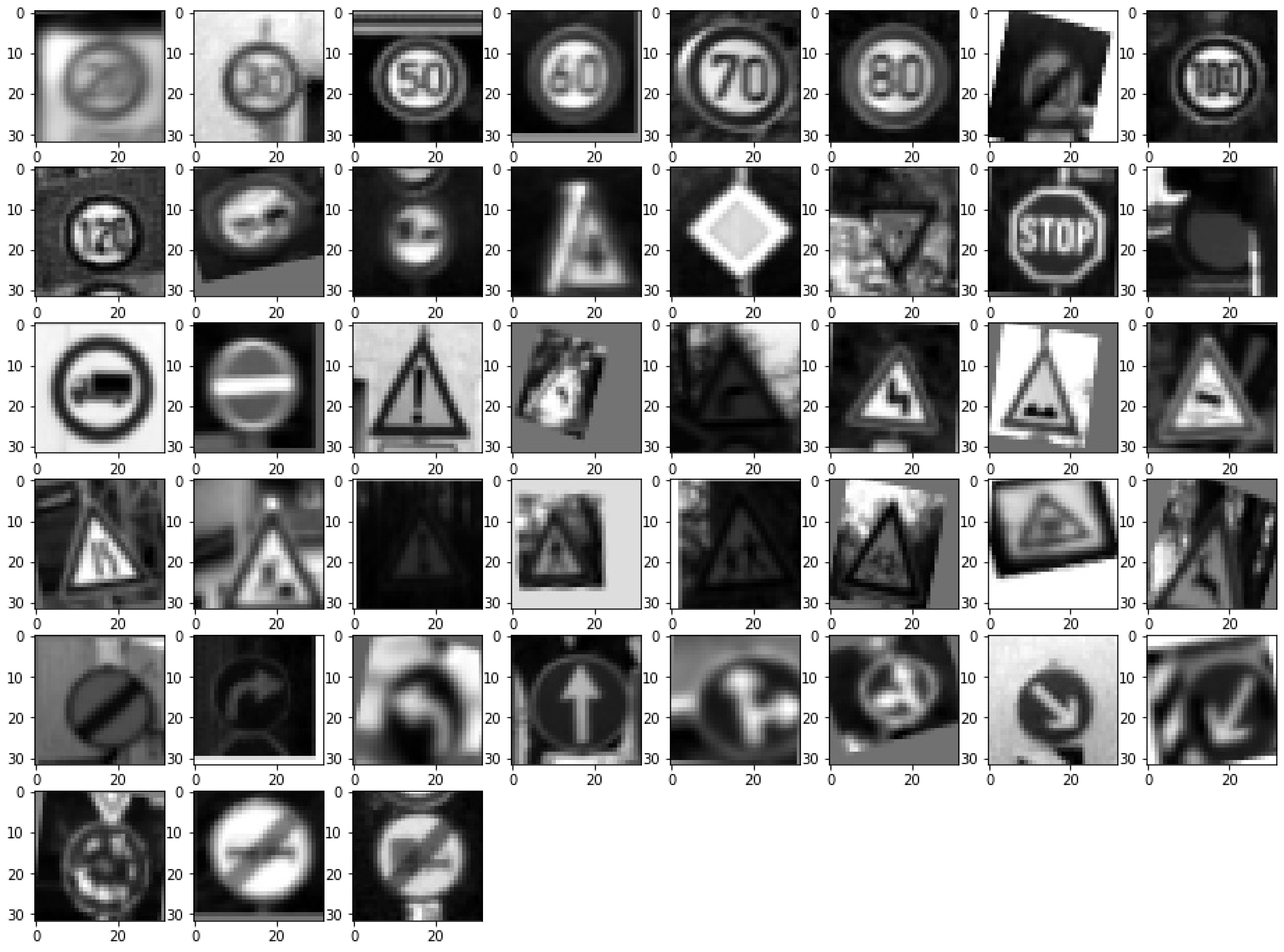
range/2 to range/2;

- rollImage: shifts the image by a random number of pixels between -range/2 to range/2;
- shearImage: applies shearing on the image between -range/2 to range/2;
- normalizeGrayscale: normalizes the gray scale image;

The cell #5 has two image processing functions, **trainingDataAugmentation** which generates new images in order to extend and balance the training data set and **inputImgPreproc** which is used to convert all images (train, validation and test) from RGB domain into gray-scale and additionally applies normalization.

At first the data has been evaluated as RGB and gray-scale, the results using gray-scale data were slightly better than the original RGB images with the advantage of only having a depth of 1. Data normalization has been applied as an effort to ensure good training convergence.

Find below some examples of images after the preprocessing stage.



There is one image of each class, they were randomly picked and one may observe that some of them are rolled while others are rotated or sheared.

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)

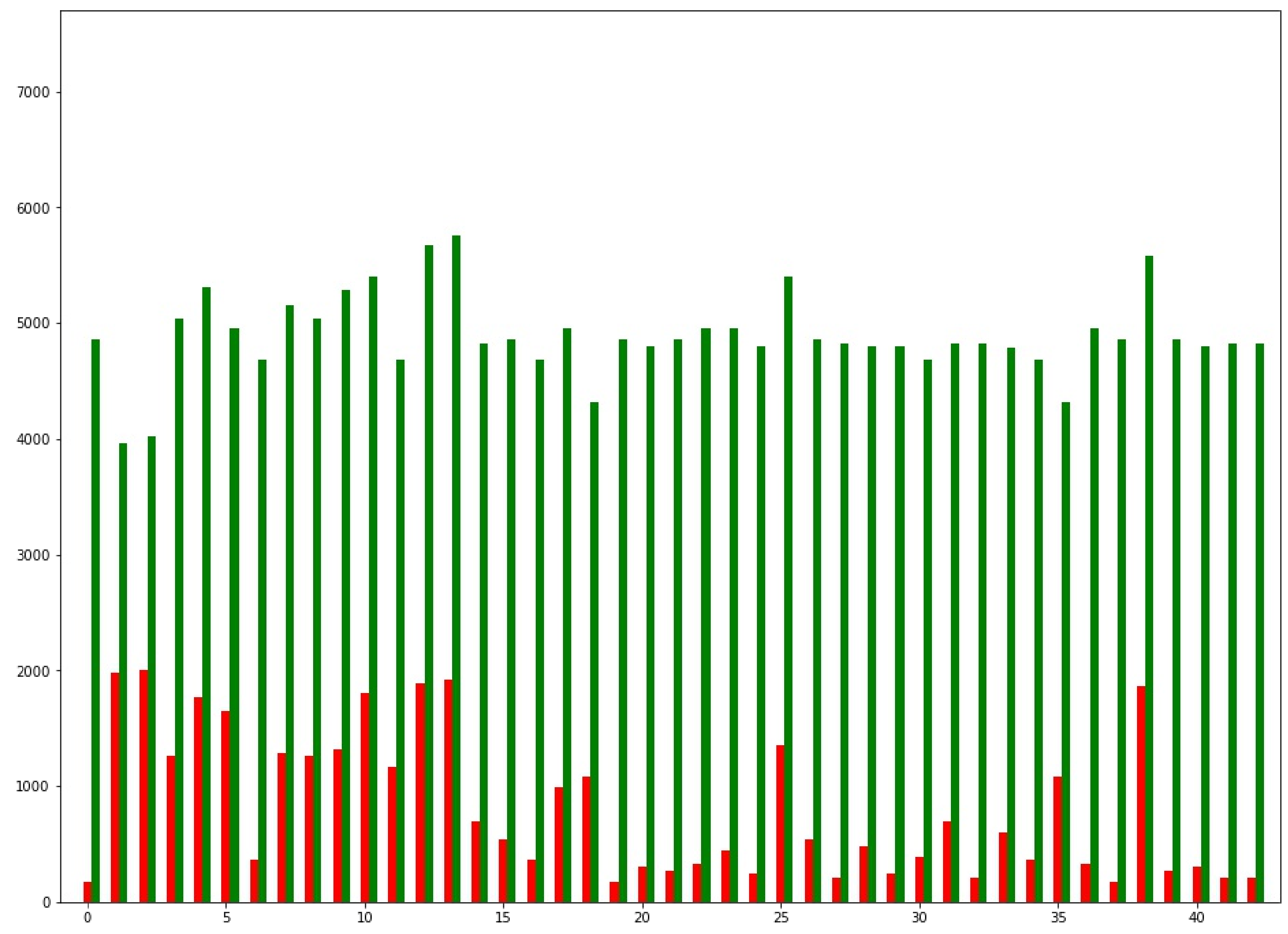
The latest version of the project on github already gave the training data set split in training and validation sets.

The training and validation are implemented on cell #14. Cell #15 has the implementation which is responsible to evaluate the trained model.

After the generation of new images, the **training** set had 210202 images, a growth of nearly 600%. The amount of data on the **validation** and **testing** data sets were kept the same (no images generated for them).

The image generation algorithm uses a mapping list which informs how many images it is supposed to generate for each image in a given class. This way the final training data set is more balanced across the different classes.

Find below a data distribution comparison red is prior to data preprocessing and green is after data preprocessing.



The main difference between the original training dataset and the augmented dataset is that the latter has artificially generated images added. This addition is done in such a way that the final dataset has approximately the same amount of training data across its classes. The generated images are the original images either rotated, rolled or sheared. All images (train, validation and test) are preprocessed with rgb2gray and normalization.

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 the final model is located cell#11. The final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grey-scale image
Convolution #1	1x1 stride, valid padding, output: 32x32x64
RELU activation	-
Max pooling	2x2 stride, outputs 16x16x64
Dropout	-
Convolution #2	1x1 stride, valid padding, output: 32x32x128
RELU activation	-
Max pooling	2x2 stride, output: 16x16x64
Dropout	-
Fully connected #0	Flatten
Fully connected #1	input: 5x5x128 output: 120
Dropout	-
Fully connected #2	input: 120 output: 84
Dropout	-
Classifier	-

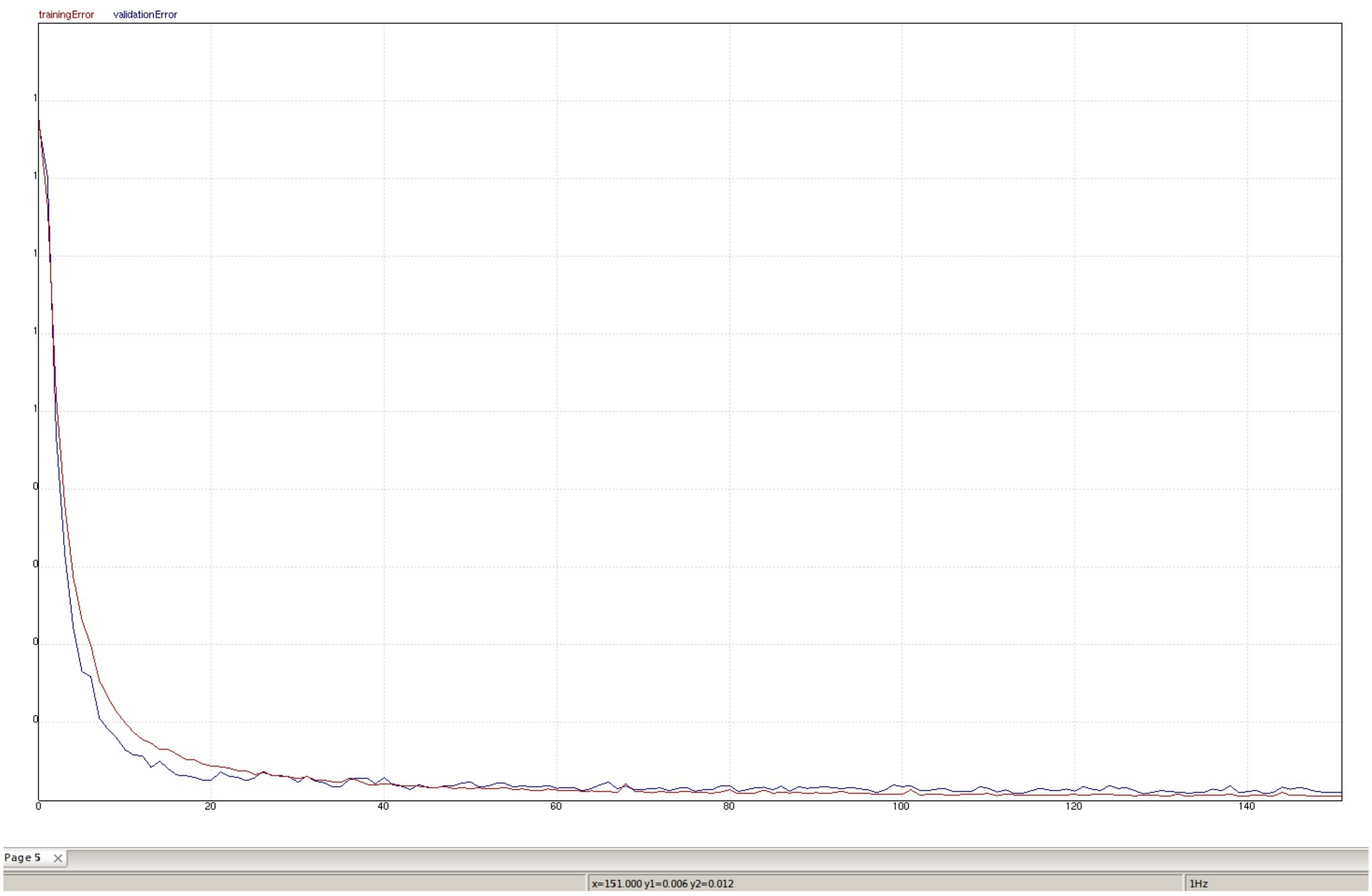
Cell #8 has some auxiliary functions to assist on the model composition.

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.

Cell #10 & #11 have the code to train the model. The optimizer selected was the Adam Optimizer. Below the hyper-parameters used on the final run.

Parameter	Final value	Description
Rate	0.001	Learning Rate
EPOCHS	150	Number of Epochs
BATCH_SIZE	128	Size of each Batch
dropout	0.5	Keep Probability for dropout function
newImages	5	Approximately the amount of new images generated per images

The image below presents the evolution of training and validation error over each epoch. After the model was able to perform better than 3% validation error, it was decided that the model was good enough to be submitted for evaluation.



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 function evaluateModel defined on cell #7 is responsible to calculate the model accuracy. The final run achieved the following results after 150 epochs:

Parameter	Value
training error	0.6%
validation error	1.15%
test error	3.1%
Processing time (NVIDIA GTX960 4GB)	4287s

The approach used to achieve this solution was the following:

- Firstly the original data was fed to the same model used to solve the LeNet problem, as suggested on the project instructions;
- It has been observed that there was much space left for improvement and then we have inserted data augmentation together with image generation;
- After that we observed that the model was overfitting the training dataset and therefore we have introduced dropout, which 'forces' the network to learn how to understand some feature even with the

absence of some of its characteristics;
- At first dropout didn't help much, several configurations/keep_prob have been attempted until some improvement has been observed (dropout between each layer);
- Dropout has improved the model accuracy but it was still overfitting the training dataset so it was a good indication that perhaps a wider/deeper network was necessary to solve the problem;
- Several architectures have been attempted, three convolutional layers approach proved to become very expensive in terms of processing time while wider networks proved to be faster and to generate better results. Most likely a combination of the two would result in an optimal solution however given the available resources it has been decided to go ahead with the wider network approach.
- A few configurations have been attempted, when the model was able to perform better than 3% of error on the validation dataset it has been decided that it was accurate enough to proceed with the testing and to submit the solution;
- That being said, the model described earlier in this report is the resulting architecture of the above mentioned approach. The final model is basically the LeNet model with wider convolutional layers and dropout.

###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.

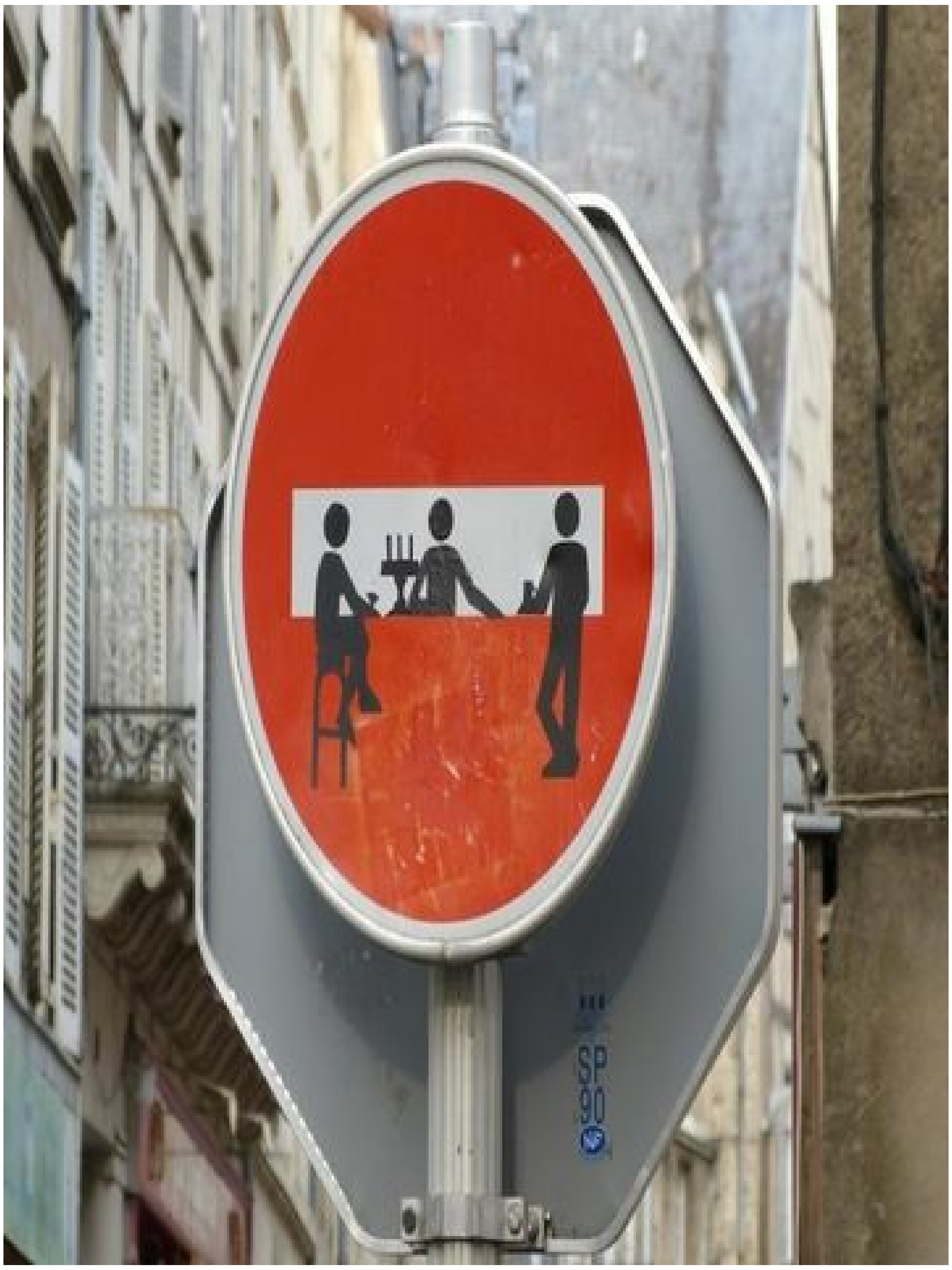
Here are eight German traffic signs which have been downloaded mainly from http://www.gettyimages.de

60
gettyimages
rolfo
459381091

STOP
HAMMER
TIME
© Various

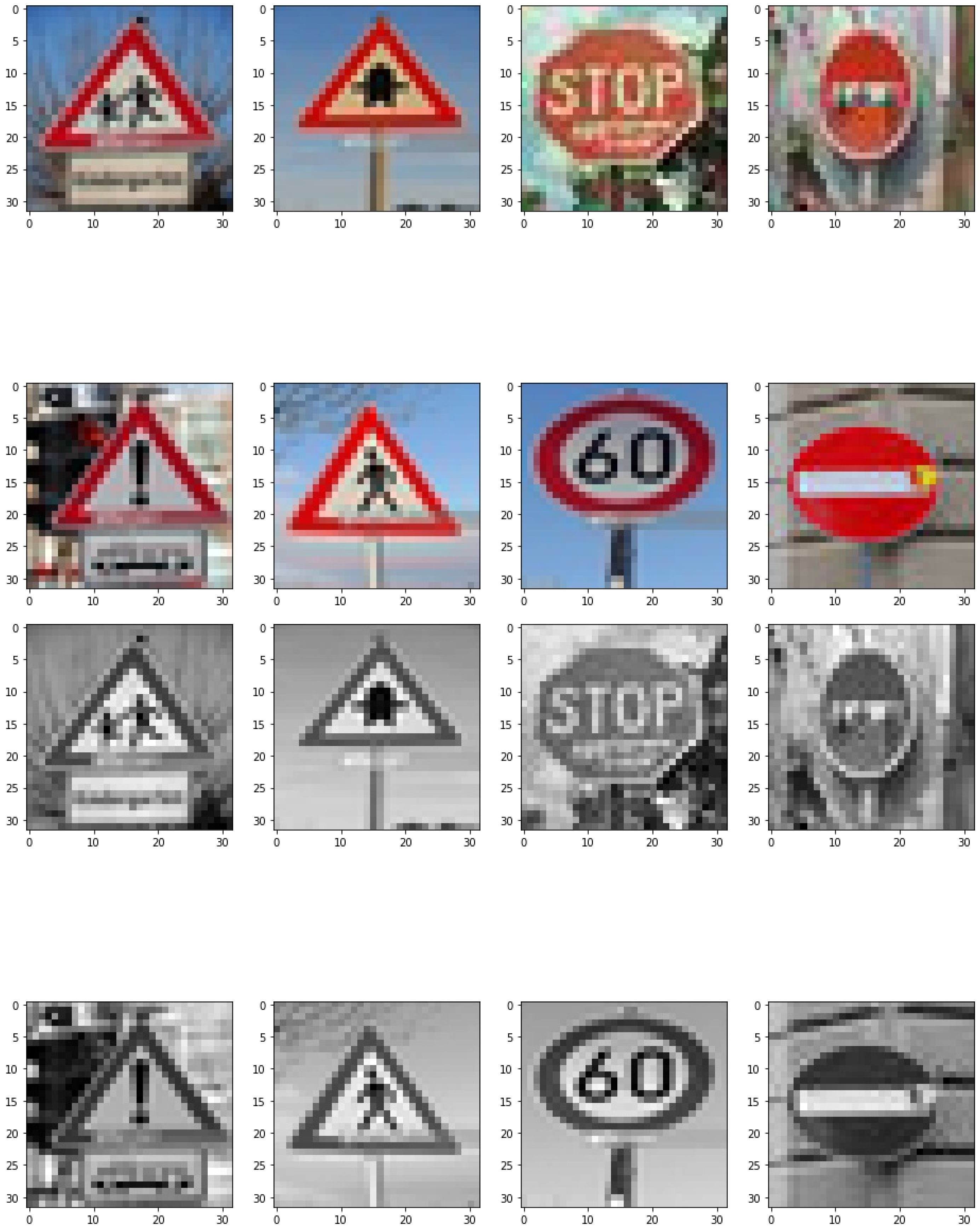
gettyimages
Rolfo Rolf Brenner
465921879

Kindergarten



Most of the images had its credits stamped somewhere over the traffic sign, so it is already a good point to observe if the model can simply ignore this 'noise' or if it will cause classification issues. The 'general caution' image had a semaphore in the background and additionally a sign below indicating a Tram. The first two images have a clear sky in the background which added high contrast between the background and the traffic sign. The stop sign image had some text added to the sign and could be problematic to classify. The 'children crossing' sign had a warning below which could also generate trouble for the model. Finally the last two images were somewhat modified from its original configuration which may mislead the model to wrongly identify features.

Below are the images after being resized to (32x32x3) and after the preprocessing stage respectively



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 trained model was able to correctly classify 7 out of the 8 selected images and therefore the accuracy was 87.5% for the selected images. Obviously these are good samples and we decided to stick with these since we already have struggled with some distorted/shaded/tilted images from the training dataset.

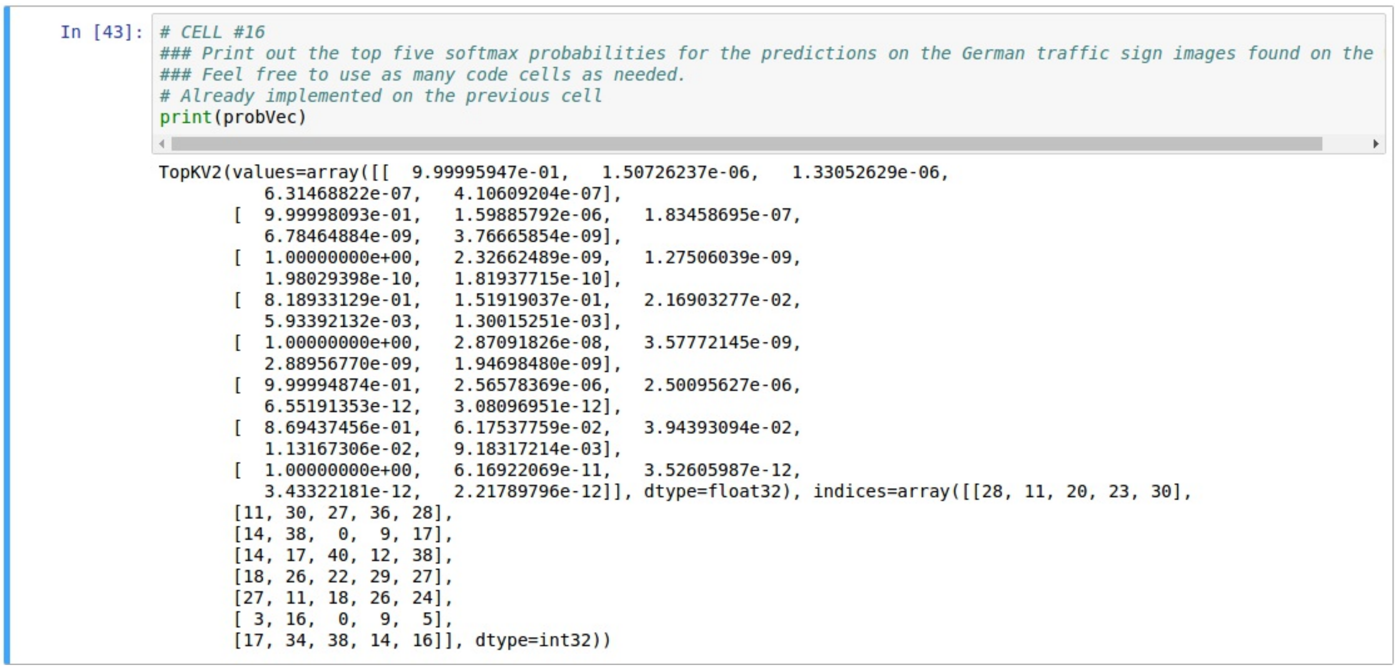
The code for making predictions on my final model is located in cell #14 & #15 of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
Children crossing	Children crossing
Right-of-way at the next intersection	Right-of-way at the next intersection
Stop	Stop
No entry graffiti	Stop
General caution	General caution
Pedestrians	Pedestrians
Speed limit (60km/h)	Speed limit (60km/h)
No entry pack-man	No entry

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)

See below the top 5 probabilities for each of the 8 images.



```
In [43]: # CELL #16
         ### Print out the top five softmax probabilities for the predictions on the German traffic sign images found on the
         ### Feel free to use as many code cells as needed.
         # Already implemented on the previous cell
         print(probVec)
```

```
TopKV2(values=array([[  9.99995947e-01,   1.50726237e-06,   1.33052629e-06,
          6.31468822e-07,   4.10609204e-07],
       [  9.99998093e-01,   1.59885792e-06,   1.83458695e-07,
          6.78464884e-09,   3.76665854e-09],
       [  1.00000000e+00,   2.32662489e-09,   1.27506039e-09,
          1.98029398e-10,   1.81937715e-10],
       [  8.18933129e-01,   1.51919037e-01,   2.16903277e-02,
          5.93392132e-03,   1.30015251e-03],
       [  1.00000000e+00,   2.87091826e-08,   3.57772145e-09,
          2.88956770e-09,   1.94698480e-09],
       [  9.99994874e-01,   2.56578369e-06,   2.50095627e-06,
          6.55191353e-12,   3.08096951e-12],
       [  8.69437456e-01,   6.17537759e-02,   3.94393094e-02,
          1.13167306e-02,   9.18317214e-03],
       [  1.00000000e+00,   6.16922069e-11,   3.52605987e-12,
          3.43322181e-12,   2.21789796e-12]], dtype=float32), indices=array([[28, 11, 20, 23, 30],
       [11, 30, 27, 36, 28],
       [14, 38,  0,  9, 17],
       [14, 17, 40, 12, 38],
       [18, 26, 22, 29, 27],
       [27, 11, 18, 26, 24],
       [ 3, 16,  0,  9,  5],
       [17, 34, 38, 14, 16]], dtype=int32))
```

Observe that for most images which the model has correctly identified, the probability/confidence is close to 100% for the first prediction and roughly zero for the other predictions meaning that the model was pretty sure about the classification in those instances. For the fourth image 'No entry graffiti' the model wasn't very sure about the prediction and the probability was only around 81% being the second 'guess' the correct prediction with 15%. The model was somewhat uncertain about the 7th image being the first prediction the correct one with probability around 87% and second option with only 6%. I believe we can say that the model performs well for new images but there is still room for improvement.