Traffic Sign Recognition

Writeup

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.

1. Files submitted, from everything I read a HTML file, notebook, and write up file is required. So, this will meet requirements.
2. Dataset summary & visualization, the rubric is referring to explaining the size of the dataset and shape of the data therein. It also would like visual explorations.
3. Design & test model: Normalized data helps Gradient Descent (or similar algorithms) to converge quickly. Hence, data normalization is one of the key preprocessing steps when it comes to data preprocessing. I experimented with preprocessing techniques (namely, MinMax Scaling and Mean Normalization). But during the initial stage of the project, I found out that, MinMax Scaling works better than Mean Normalization for this dataset. One-Hot Encoding was used to convert label numbers to vectors. A small percentage of training data (indicated by dev_data_percentage) was used as my development set. Other important functions were defined in impfunctions.py

4. Test model on new images, I found new images on the internet and tried to find images that were already classified out of the 43 classes. After testing on the new images it was found that the model even though performed sensibly in classifying them, the accuracy was at 20%

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.

Number of training examples = 39209

Number of testing examples = 12630

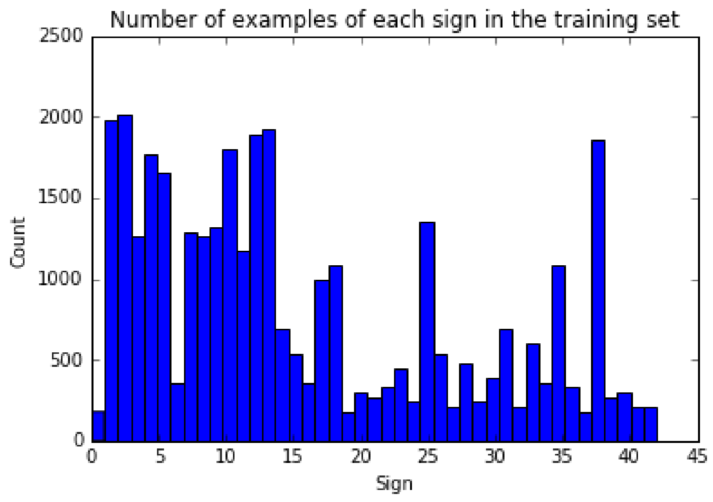
Image data shape = (39209, 32, 32, 3)

Number of classes = 43

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

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data ...

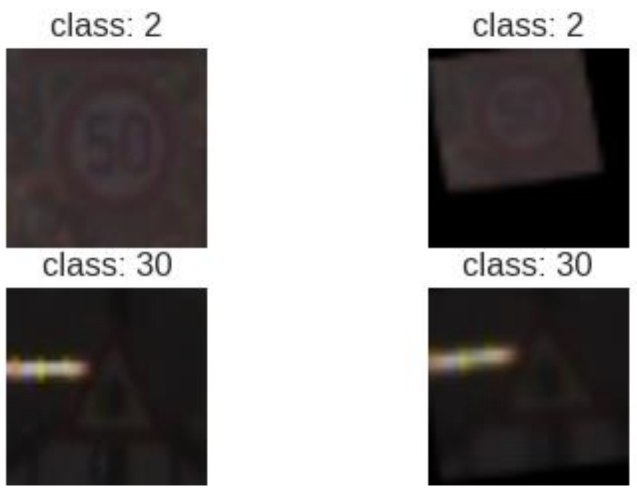


Design and Test a Model Architecture

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. When it comes to training a neural network cross-validation dataset help us to evaluate the performance of the model without using the final testing dataset. I used 1 percent of the original dataset as my dev dataset. (this dataset consists of 39209 training examples. Hence, a random 10% is a good representative sample for cross-validating my model)
2. At this point, I want to quickly design a Minimum Viable Model which help me to get a quick idea about the predictive performance. So, I used LeNet model design. In addition to that, at this stage, my model will be completely based on the training dataset only (no augmented data generated).
3. Augmented Image creation: For each image to the training dataset, small random Rotation, Translation and Shear operation was applied. So, by using this approach I was able to double my training dataset. transform_image method in python was used to generate augmented image.

Here is an example of an original image and an augmented image:



But since the accuracy dropped after using augmented images I stuck with the previous given data and ignored augmented data.

2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.)

1. When I start a new machine learning project usually I start with a very simple algorithm. That approach helps me quickly build a model and experiment with the dataset. So in this project also I used that approach and build a simple convolutional neural network as my first attempt.
2. So my first model inspired by LeNet has the following configuration.
   A. First convolutional layer: 5x5 kernel, stride of 1, depth of 6, VALID padding.
   B. Max pooling layer: 2x2 kernal with stride of 2
   C. Second convolutional layer: 5x5 kernel, stride of 1, depth of 16, VALID padding.
   D. Max pooling layer: 2x2 kernal with stride of 2
   E. Fully connected layer of size 1024
   F. Dropout layer with dropout probability 0.5
   G. Fully connected layer of size 1024
   H. Dropout layer with dropout probability 0.5
   I. Finally, a softmax layer with size 43
3. For regularization I used dropout (with probability: 0.5) and regularizer with small regularization parameter.

Architecture of my second model:

1. First convolutional layer: 3x3 kernel, stride of 1, depth of 64, SAME padding.
2. Max pooling layer: 2x2 kernal with stride of 2
3. Second convolutional layer: 3x3 kernel, stride of 1, depth of 64, SAME padding.
4. Max pooling layer: 2x2 kernal with stride of 2
5. Second convolutional layer: 3x3 kernel, stride of 1, depth of 128, SAME padding.
6. Max pooling layer: 2x2 kernal with stride of 2
7. Fully connected layer of size 1024
8. Dropout layer with dropout probability 0.5
9. Fully connected layer of size 1024
10. Dropout layer with dropout probability 0.5
11. Finally, a softmax layer with size 43

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.

First Model:

1. Optimizer: AdamOptimizer
2. Learning Rate: 0.001
3. Epochs: 100
4. Batch Size: 128
5. Regularization Parameter: 1e-6

Second Model:

1. Optimizer: AdamOptimizer
2. Learning Rate: 0.001
3. Epochs: 120
4. Batch Size: 128
5. Regularization Parameter: 1e-6

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 had test set accuracy of 96.62

I started the model building using LeNet architecture. The main reasons for selecting that model were:

- It’s well known and performs well on computer vision tasks
- Training time is not very high (took me a little more than an hour to train).
- Initially, I didn't use additional data. However, even without those additional data, I managed to get 95.42% accuracy.

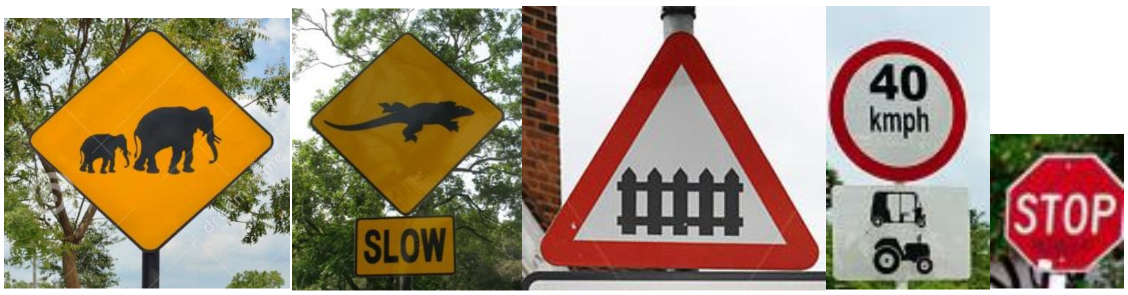
But this neural network architecture didn't work so well when I add some additional synthetic data (please look at the graphs). So I decided to use a mid-size network. The main reson for me to select a mid-size convolutional neural network was my computational budget.

Also, loss/accuracy graphs of that network suggest, there is a possibility of pushing the accuracy a little bit further if the increase the size (number of layers, computational units per layer and etc.) of the network.

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.

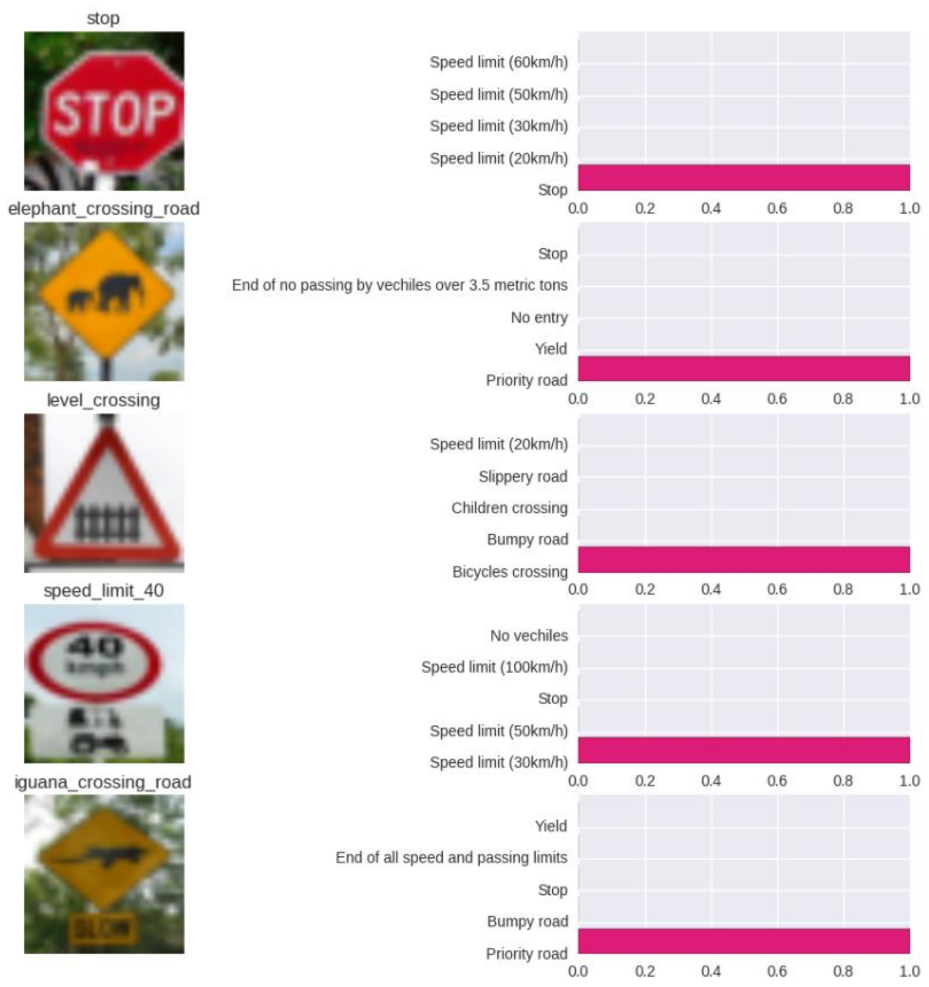
Here are five German traffic signs that I found on the web:



All images except stop sign would be difficult to classify and I used them to check the generalization capability of the model

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 1 of the 5 traffic signs, which gives an accuracy of 20%. The stop sign here was very similar to the stop sign provided in the dataset hence it was accurate. For the rest of the images the model made sensible predictions. The elephant crossing and Iguana crossing road are both classified as priority road which is correct. Also it classified level crossing as bicycle crossing which is a fair interpretation of the image.

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)

From the above visualization the code gives 100% probability for 1 stat and 0 for all the remaining 4 stats of all the new images which is confusing and needs more looking into