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.

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

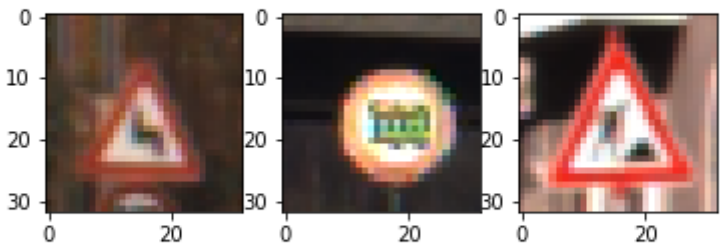
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used the pandas library, specifically `shape` function to find the required summary statistics of the traffic signs data set. For the finding the number of unique classes, I used numpy's `unique()` function on the given labels.

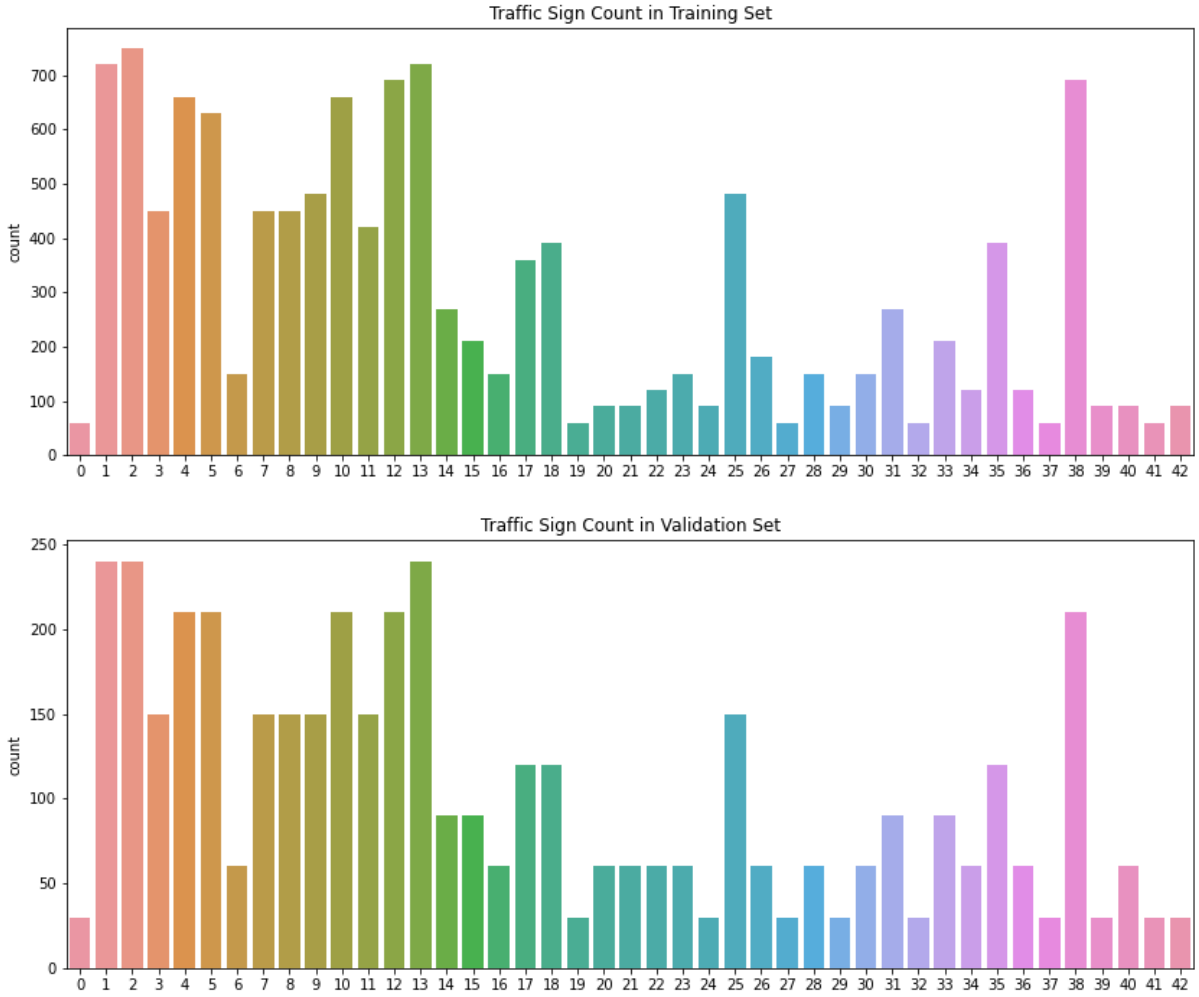
- The size of training set is **34799**
- The size of the validation set is **4410**
- The size of test set is **12630**
- The shape of a traffic sign image is **32x32x3**
- The number of unique classes/labels in the data set is **43**

2. Include an exploratory visualization of the dataset.

Each image on the dataset has 32x32 size for height and width. Each image is also given in color, which means each image has depth of 3. Below, you see couple of exmaples from the dataset.



I've also looked at how many of each traffic sign exists in the dataset. I thought the best way of visualizing it was through a bar chart. Below, you see the labels for training and validation set.



The bar plots show us that the types of traffic signs are not evenly distributed; however, training and validation sets have a very similar distribution.

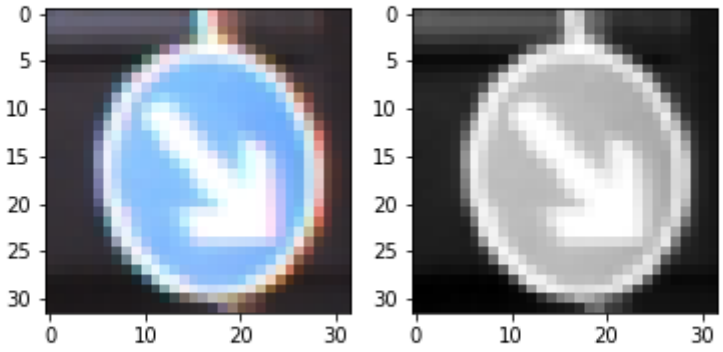
The visuals above gave me enough understanding of the dataset, therefore I have not included any additional visualizations.

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.)

Although traffic signs utilize color, it is typically not the main property. For example, a stop sign is universally red; however, the more identifying feature is the text **STOP** rather than the color. Therefore, I decided to exclude color information by converting the images to grayscale. Below, you see an example of an image before and after grayscaling.



The last step of preprocessing was normalization of the data. Normalized data sets are recommended as it improves training of the model by having both positive and negative values and it helps the model to converge quicker.

For preprocessing, I also spent quite a bit of time in Udacity knowledge base, especially when I first started the project and I realized the model was overfitting. I tried to understand what is meant by augmentation, which was a common discussion point. At first, I thought augmentation was *only modifying* the existing training data, but not *extending*. This generated a lot of confusion to me. Now, I understand that this is a mechanism to *balance* the training data, as training data didn't have even distribution among the traffic signs. However, as I met the required accuracy for the project via modifications to the model, I didn't try further augmentation.

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 model follows the LeNet architecture from the classroom lab.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 grayscale normalized image
Convolution 3x3	1x1 stride, same padding, outputs 28x28x6
RELU	Activation Layer
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	1x1 stride, same padding, outputs 10x10x16
RELU	Activation layer
Max pooling	2x2 stride, outputs 5x5x16
Fully connected	Input 400 to Output 120
RELU	Activation Layer
Dropout	Keep probability = 0.5
Fully connected	Input 120 to Output 43
Softmax	

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.

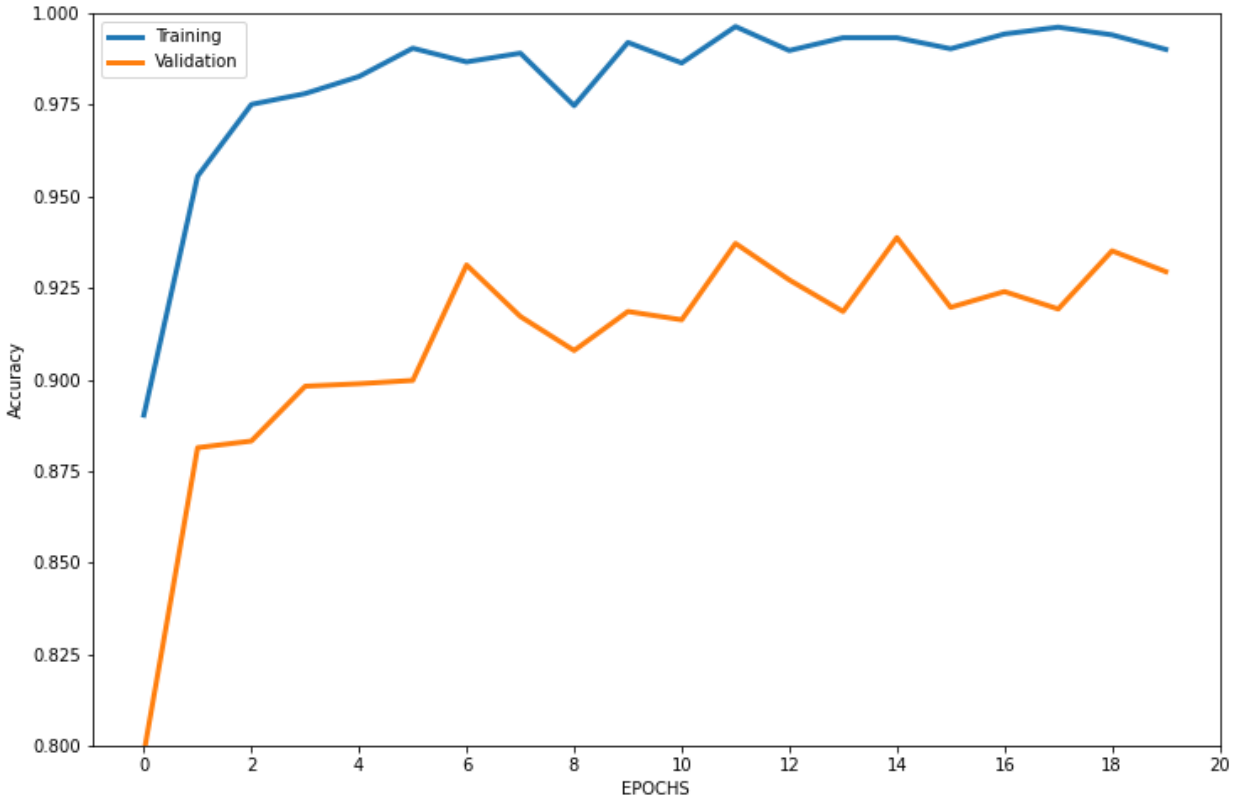
I referred to the course material again for training the model and used the Adam optimizer, as this was suggested as a good starting point for optimizer in the lectures. In the beginning, I set the number of epochs to 10, the batch size to 128 and the learning rate to 0.001, same as the lab. However, I changed these to 20, 256 and 0.005 respectively.

First of all, it didn't seem like the model finished converging when I set the epochs to 10, therefore I increased it to 20. I also thought that the batch size was too small, as there are 43 types of traffic signs and the counts are not even. Now that the batch size is increased, increasing the learning rate also

made sense as the model will be trained on more samples and it makes sense to take a *larger* step with each batch.

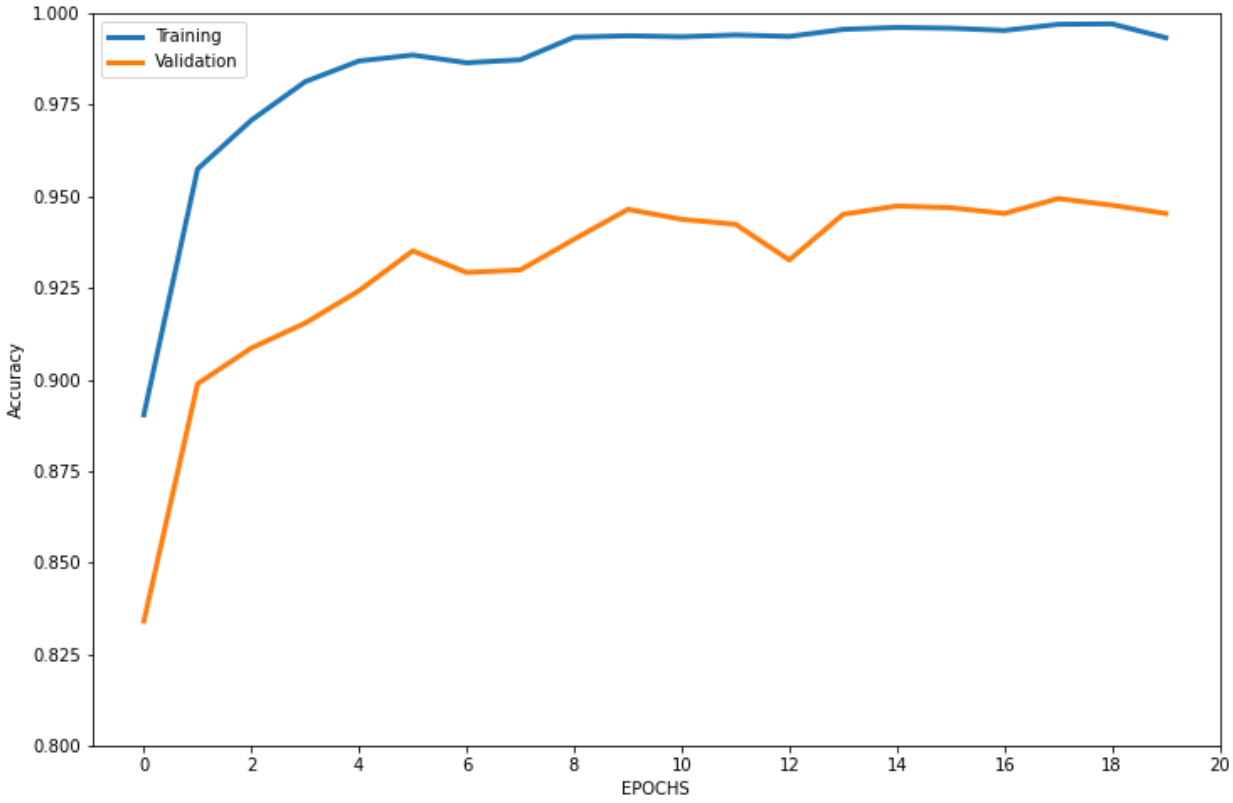
4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, 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.

For this project, I started with the LeNet architecture from the coursework. The main reason for choosing LeNet is because it was encouraged as a good starting point, and it performed fairly well in the course Lab. However, the model was overfitting, as the training accuracy was 0.99 or higher, while the validation accuracy couldn't reach 0.93.



LeNet architecture consists of 2 convolutional layers, each followed with max pooling, and then 3 fully connected layers. As the model was overfitting, I thought it made sense to remove one of the

connected layers to reduce the model parameters. In addition, the dataset has uneven distribution of samples. Therefore, I thought a dropout layer made sense between the fully connected layers as a measure to prevent overfitting. I tried 3 drop out rates: 0.80, 0.75 and 0.50. Among these 0.50 gave me the best validation accuracy. The results can be seen below.



As it can be seen from the plot, the model is *still overfitting*. I believe this can be further mitigated by augmenting the training data, by creating new images. However, as mentioned above, I didn't pursue this route for this project. In the course material, it was also mentioned that LeNet was created to solve a simpler problem. I thought about adding more convolutional layers to make it deeper; however, the results were already satisfying. Therefore, I didn't add any additional layers to the model.

My final model results were:

- training set accuracy of **0.993**
- validation set accuracy of **0.945**
- test set accuracy of **0.928**

The final values for the model indicates that the model is still overfitting, as the difference between training and test set is still significant. Nevertheless, I think the overall performance of the model is very good as test set also contains a significant number of samples.

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:

The images are collected from various sources from image search websites, respectively they are Speed Limit (50 km/h), Bicycle Crossing, No Passing, Road narrows on the right, and slippery road.

I expect the model to have difficulties with slippery road and bicycle signs, as the shapes are harder to understand in the 32x32 images, even for human eye. I'm not sure if the convolutional neural network is deep enough to identify all the features of a bicycle to be able to form the whole picture and classify correctly. Similarly to slippery road sign, the squibbly lines below the vehicle may not be well identified; however, this type of road sign is fairly unique too.

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:

Image	Prediction
Speed Limit (50 km/h)	Speed Limit (50 km/h)
Bicycle	Roundabout mandatory
No Passing	No Passing

Image	Prediction
Road Narrows on the Right	Road Narrows on the Right
Slippery Road	Slippery Road

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. The size of the dataset is very small; however, the accuracy is quite lower than the test accuracy of the original dataset. This could've been easily different if different traffic signs with easier to detect features were selected for testing. Only bicycle was classiffied incorrectly, which is somewhat expected due to the complexity of features it contains.

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 the following:

```
softmaxOut = tf.nn.softmax(logits)
topProbs = tf.nn.top_k(softmaxOut, k=5)

with tf.Session() as sess:
    saver.restore(sess, './myModelTS')

    probs = sess.run(topProbs, feed_dict = {x: X_test_new, keep_prob: 1.0})
```

For the first (speed limit 50 km/h) and third image (no passing), the model was very confident to classify these with probability very close to 1.0. And these were correctly classified. Similarly, the fifth image (slippery road) also had a probability very close to 1.0. Therefore, I won't list the top 5 probabilities for these images, as the remaining 4 out of the softmax were negligible. These values are displayed in the Jupyter notebook.

For the second image, the image was classified incorrectly. The model classified this as a speed limit (30 km/h) with a probability of 0.74, and this was incorrectly classified. On top of it, none of the top 5 probabilities were of bicycle, so the model performed very poorly on this image. The top five soft max probabilities were:

Probability	Prediction
.74	Speed limit (30km/h)
.18	Speed limit (100km/h)
.05	Speed limit (120km/h)
.02	Go straight or left
.01	Roundabout mandatory

For the fourth image, the image was classified correctly as 'Road Narrows On Right" with a probability of 0.96. This was pretty good, as it was also classified correctly. However, I will still display the top 5 probabilities here to show the good performance of the model in classifying this image.

Probability	Prediction
.959	Road narrows on the right
.004	Pedestrians
1.8e-13	Children crossing
1.1e-14	Right-of-way at the next intersection
3.6e-14	Double curve