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. Basic summary of the data set.

I used the numpy library 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 32x32x3 (3 color channels).
- The number of unique classes/labels in the data set is 43.

2. Exploratory visualization of dataset

Below are class histograms for the training, validation and test data sets are displayed.

- The distributions of each subset appear nearly identical.
- Around 50% of the classes have substantially lower frequency than others.
- This could lead to higher prediction uncertainty for the under-represented classes.

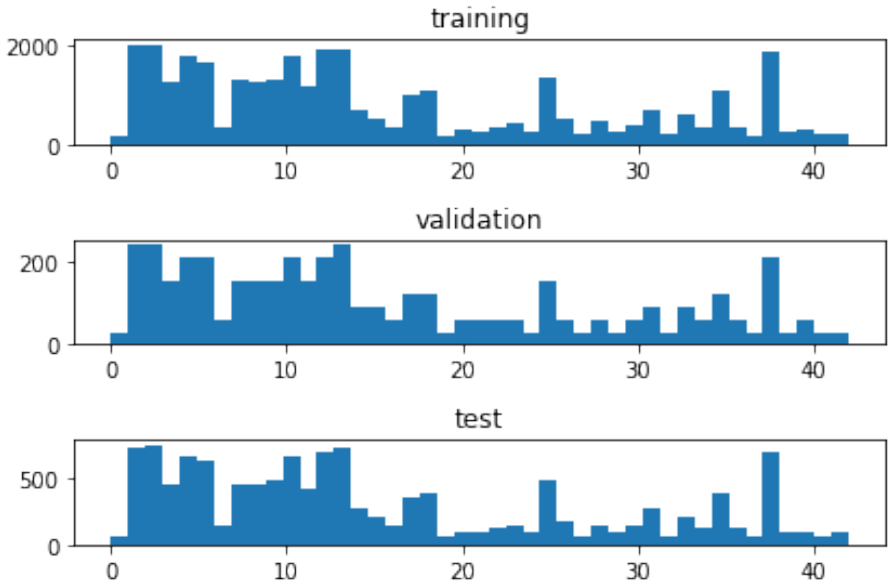

Figure 1: Class histograms

Design and Test a Model Architecture

1. Image pre-processing

Prior to training or prediction, images are pre-processed as follows:

- Images are normalized, per channel:
  - images are converted to floating-point representation
  - each channel is offset by the channel mean (mean is subtracted)
  - each channel is divided by the channel standard deviation
- The resulting pre-processed image data is approximately zero-centered, with standard deviation = 1.0.
- This technique should normalize the data for variations in image brightness, which is a major source of variation in the data set.

During the process of developing & testing my classifier, several variations on pre-processing were tried.

- Grayscale conversion
  - Images were converted to grayscale before normalization.
  - The intent in this case was to remove color variation from the inference model, as it may not contribute meaningfully to the difference between image classes.
  - In my testing I found no meaningful difference in predictive power (validation set accuracy) between using color images and grayscale converted images. Both methods resulted in validation set accuracy around 96%.
- Random cropping
  - Images were cropped to 24x24 before training & inference
  - During training the images were randomly cropped to a 24x24 region of the training image.
    - Using this method, there was no fixed training set, but rather a continuous stream of randomly cropped images used for training the model.
    - The intent was to make the model more robust to translational variation and occlusions (where part of the sign would fall outside the cropped region). I believed this would reduce over-fitting to the original training data.
  - During validation & testing, images were cropped to the central 24x24 region of the image for consistency.
  - I found that this method did improve prediction accuracy by around 1%, but at the cost of much slower training.
    - Without cropping, the model reached 96% accuracy after 20 training iterations.
    - With cropping, the model reached 97% accuracy after 150 training iterations.
- Random cropping with grayscale conversion
  - Images were converted to grayscale, then applied the cropping scheme described above.
  - As before, I observed no meaningful performance difference with or without grayscale conversion.

There are a few methods of pre-processing that I considered, but did not apply due to time constraints:

- For the following, I would favor using the same technique as I did with random cropping–applying a randomized set of image transformations "on the fly" to each training image. This should provide effective in reducing over-fitting and increasing robustness to many variations present in real-world images.
- Random image scaling
  - Re-size (and crop) training images within +/- 50%

  - Account for "seeing" signs from various distances (very important)
- Random perspective distortion
  - Simulate rotating the image around vertical axis within +/- 45 degrees
  - Account for "seeing" signs from various angles (very important)
- Small rotations
  - Rotate training images within +/- 15 degrees
  - Account for "seeing" signs that are moderately rotated
  - Some traffic sign meanings are orientation sensitive, so larger rotations would confuse the model
- Add random image noise

Some other popular methods for augmenting training images would not be effective for traffic sign recognition, such as:

- Vertical and/or horizontal flipping – can change meaning of sign
- Large rotations – can change meaning of sign

2. Model Architecture

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x32
RELU	
Max pooling	2x2 stride, outputs 14x14x32
Convolution 5x5	1x1 stride, same padding, outputs 14x14x64
RELU	
Max pooling	2x2 stride, outputs 7x7x64
Flatten	Inputs 7x7x64, outputs 3136 (flat)
Fully connected	Outputs 1024
RELU	
Fully connected	Outputs 512
RELU	
Fully connected	Outputs 256
Softmax	

The model includes options for enabling L2 Regularization for all weights, and Dropout in the fully connected layers. I observed both to be useful methods for reducing over-fitting.

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 used mini-batch gradient descent with the "Adam" optimizer to train the model.

- I used a batch size of 128. Also tried 256, but there was no noticeable difference.
- I used a learning rate of 0.0003 and trained for 20 epochs. The training set was re-shuffled on each epoch.
- I tried many learning rates, varying between 0.00005 and 0.1. The rate of 0.0003 struck a good balance between model accuracy and training time.

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:

- training set accuracy of 0.998
- validation set accuracy of 0.966
- test set accuracy of 0.960

I applied an iterative approach to designing the model architecture:

- I started with the LeNet architecture we learned in the class lesson. The initial model–without any features for reducing over-fitting–achieved around 90% validation accuracy. Not very good.
- After adding features like dropout and L2 regularization, various image pre-processing steps, and tuning the learning rate, the model topped out around 92-93% validation accuracy. I didn't think that was nearly good enough.
- I studied some example architecures on the TensorFlow website for classifying the MNIST and CIFAR-10 image sets, which followed a similar pattern to LeNet: 2 convolutional layers, followed by a series of fully connected layers. However, I noticed that these example models utilized far more parameters in the convolutional layers, when compared to LeNet.
  - I tried adding a similar number of parameters in the 2 convolutional layers: 32 channels in the 1st layer, and 64 in the 2nd. This improved model accuracy to about 94%.
  - I noticed during training that the gap between training accuracy and validation accuracy was still pretty large, which I think indicates

excessive over-fitting. I tried adding dropout between each of the fully connected layers, instead of only on the last (read-out) layer.
- The increased dropout resulted in the validation accuracy increasing to a consistent 95-96%.
- I tried dropout rates between 0.3 and 0.5, and settled on 0.4. Dropout rate below 0.4 seemed to under-fit the model.
- I then tried enabling L2 regularization (weight decay). I tried many decay rates, most of which resulted in under-fitting the model. But decay rates around 0.002 to 0.003 resulted in model improvement to 96-97% validation accuracy.

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:

Reasons why these images might be difficult to classify:

- When reduced to 32x32 the bicycle shape is hard to distinguish, and all the caution signs have the same shape & colors.
- The ice symbol looks similar to the right-of-way symbol (similar sign), and all the caution signs have the same shape & colors.

- The no passing sign should not be difficult–it's clearly distinguished from other signs.
- The single pedestrian symbol looks similar at 32x32 to the general caution symbol, and all the caution signs have the same shape & colors.
- The speed limit 30 sign should not be difficult at all.

These images were not selected to be difficult, in fact they were selected to be reasonably easy for the model to predict. In fact, selecting images that could be correctly classified was quite a learning exercise in itself.

When I first started collecting images for this part of the assignment, I intentionally chose examples that seemed like they could be challenging. That proved to be true–my trained model was unable to correctly classify any of the images I had chosen. Next I chose several more images that I thought should be fairly easy to classify. Unlike my first batch, these signs were well lit, oriented perfectly, with excellent contrast. I found that my model was still unable to classify any of them correctly!

At that point, I coded up a script to extract all the training images from the

pickle file archive, and actually browsed through many of the images. Obviously I should have done this sooner! Some of my discoveries:

- For each sign class, the training image signs are uniformly sized, very well centered, and take up most of the image.
- Most of the variation in the training set seems to be brightness variation, which is greatly reduced pre-training by image normalization.
- Some German traffic signs (bicycle crossing, pedestrian crossing, etc.) come in a variety of colors, shapes and symbol designs. However, the training images for these classes are uniform in sign design.
- Sure enough, any image I selected where the sign was significantly than in training images, or used any kind of variation in design, were unrecognizable by my trained model.

With this in mind, I purposefully selected example images that were at least somewhat good candidates to be recognized by the model. What I really learned through this exercise is the importance of assembling a representative training set, and applying a purposeful variety of image distortions during training.

Specifically for the traffic sign classifier, the training images should encompass:

- Examples of each sign design in use for a given class (including variations in color, shape, and symbols).
- A variety of sizes for each sign–this can be artificially simulated with random re-sizing (as described in the image pre-processing section).
- Other randomized distortions like translation, perspective changes, small rotations, etc.

2. Predictions on the new images

Here are the results of the prediction:

Image	Prediction
Bicycle Crossing	Bicycle Crossing
Beward ice/snow	Beware ice/snow
No passing	No passing
Pedestrians	Right-of-way next intersection
Speed limit 30	Speed limit 30

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This is a reasonable result given the model test accuracy of 96% on a much larger test set.

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each pre-

diction. 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 11th cell of the Ipython notebook.

For the first image, the model is moderately certain that this is a "Bicycle crossing" sign (probability 66.9%), which is a correct classification. The top five soft max probabilities were

Probability	Prediction
.669	Bicycle crossing
.130	Beware ice/snow
.087	Children crossing
.055	Road work
.014	Bumpy road

For the second image, the model is moderately certain this is a "Beware of ice/snow" sign (probability 76.9%), which is a correct classification. The top five soft max probabilities were

Probability	Prediction
.769	Beware ice/snow
.224	Right-of-way next intersection
.006	Speed Limit 100
.0003	Double curve
.0001	Road work

For the third image, the model is highly certain this is a "No passing" sign (probability 99.9%), which is a correct classification. The top five soft max probabilities were

Probability	Prediction
.999	No passing
.0008	No passing for vehicles over 3.5 tons
.0001	No vehicles
.0001	Vehicles over 3.5 tons prohibited
.0001	End of no passing

For the fourth image, the model is weakly certain this is a "Right-of-way next

intersection" sign (probability 52.9%), which is incorrect. The correct class "Pedestrian" is the #2 prediction (probability 41.8%). The top five soft max probabilities were

Probability	Prediction
.529	Right-of-way next intersection
.418	Pedestrians
.044	General caution
.0055	Double curve
.0022	Road narrows on the right-of-way

For the fifth image, the model is highly certain this is a "Speed Limit 30" sign (probability 99.9%), which is a correct classification. The top five soft max probabilities were

Probability	Prediction
.999	Speed Limit 30
.0001	Speed Limit 20
.0000	Speed Limit 70
.0000	Speed Limit 50
.0000	Speed Limit 80