

Traffic Sign Recognition

Writeup Template

By ~Karanj

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

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 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.

Cell 1: Loading Libraries

Cell 2: Loading DataSets

Used extra DataSet "trainpp_brtNormAug.p", which includes 34799 original train images and another 34799 preprocessed images.

Cell 3: (Some of them are included in Cell 2)

- myX_train and myy_train are from DataSet "trainpp_brtNormAug.p"
Image Shape -myX_train: (32, 32, 3)

Training Set -myX_train: 69598 samples

[36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36]

Image Shape: (32, 32, 3)

Training Set: 34799 samples

Validation Set: 4410 samples

Test Set: 12630 samples

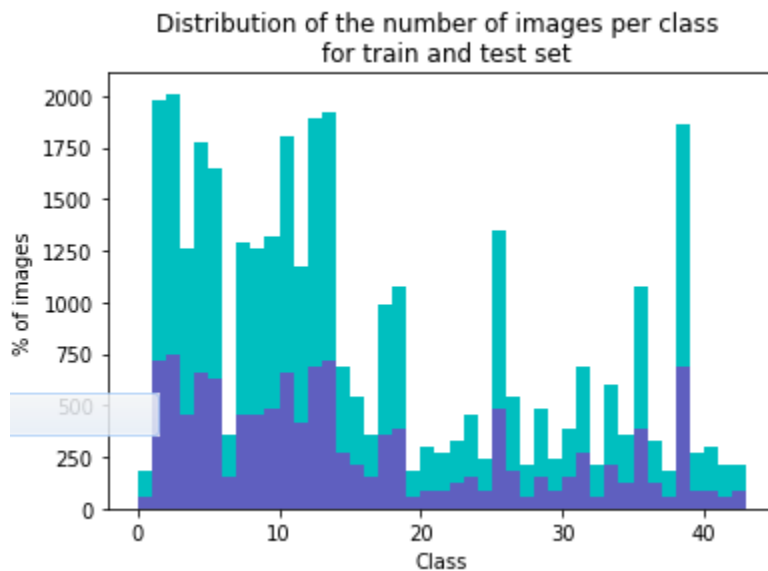
Image data shape = (32, 32, 3)

Number of classes = 43

2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

Cell 4: Random 25 images with their sign names from training data set

Cell 5: Train set color: Cyan | Test set color: Magenta



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.

Cell 6: def `brightness_process_image`(Xpp_in, Xpp_out):

- Brightness Normalization using `cv2.normalize`
- Brightness Augmentation

Cell 7: def `transform_image`(image,ang_range,shear_range,trans_range):

- Rotation
- Translation
- Shift shear
- Call Cell 6 function

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)

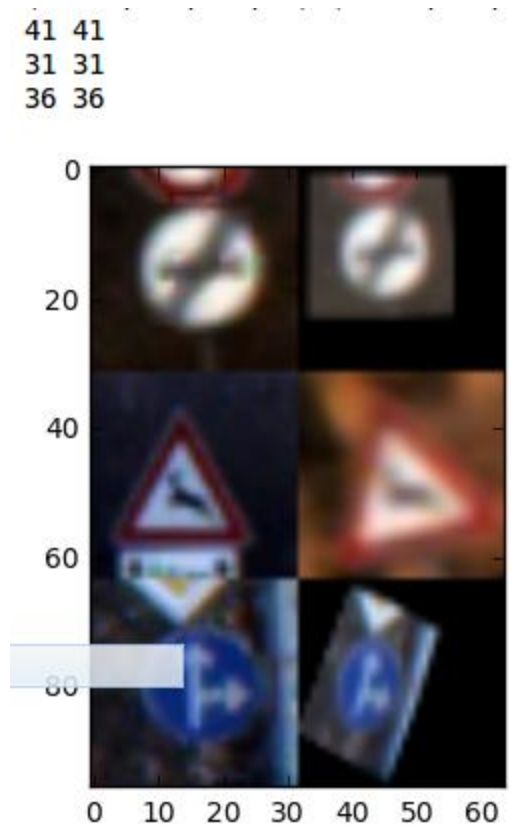
Cell 8: def `preprocess_images`(Xbtin_train, ybtin_train, Xbto_train, ybto_train):

- Call Cell 7 function
- Append/ Concatenate newly generated images to given base train dataset
- Create input/output arrays for new dataset

Cell 9: Call Cell 8 function

- write this 'trainpp_brtNormAug.p' file which includes base images plus preprocessed images. Used `pickle.dump` to write new file
- **Total 69598 images (Double of base training DataSet)**
- Print some results of pre-processing done in Cell 8,7,6
- Input training dataset shape: (34799, 32, 32, 3)
- Output training dataset shape: (69598, 32, 32, 3)

Cell 10: Randomly see any three: Original image | Preprocessed image



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.

Cell 11: Epochs, Batch_Size, keep_probability
 Cell 12: Model Implementation :

Layer	Description [Cell 12]
Layer 0	Input: 32x32x3 RGB image. Output : 32x32x3
Process L0	1x1 pixel convolution (twice, model learns color depth)
Layer 1	I: 32x32x3 -> 30x30x6 -> O: 15x15x6
Process L1	Convolution, Relu, max_pool
Layer 2	I: 15x15x6 -> 13x13x32 -> O: 6x6x32

Process L2	Convolution, Relu, max_pool, dropout
Layer 3	I: 6x6x32 -> 4x4x64 -> O: 4x4x64
Process L3	Convolution, Relu, max_pool, dropout
Flatten	Input = 4x4x64. Output = 1024.
Layer 4	Input: 1024 -> Output : 256
Process L4	Fully Connected, forward GD, dropout
Layer 5	I: 256 -> O: 128
Process L5	Fully Connected, forward GD, dropout
Layer 6	I: 128 -> O: 43
Process L6	Fully Connected, forward GD

Cell 13: Tf.placeholders

4. Describe how, and identify where in your code, you trained your model. The discussion can include the type of optimizer, the batch size(cell 11), number of epochs(cell 11) and any hyperparameters such as learning rate.

Cell 14: Beta for regularization (hyperparameter)

Cell 15: Call Model in Cell 12

- cross_entropy
- loss_operation
- optimizer
- training_operation

Cell 16: Model evaluation

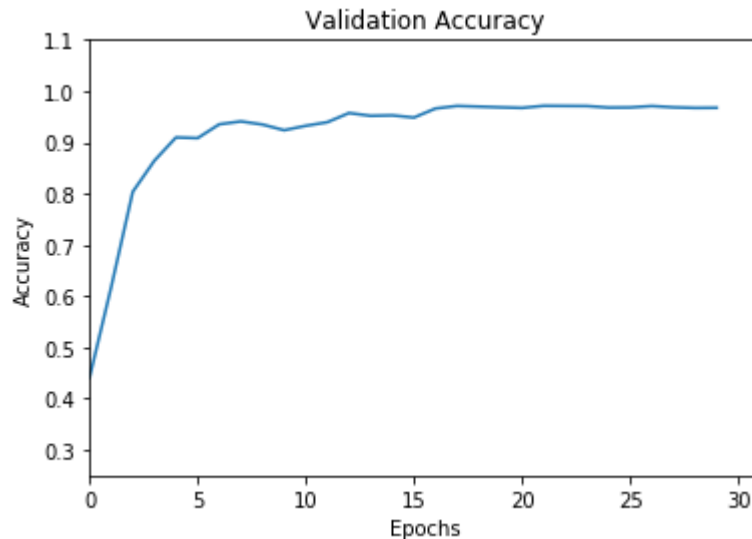
- correct_prediction
- accuracy_operation
- def [evaluate](#)(X_data, y_data):

Cell 17: Train Model and Print Accuracy for Validation set and [Save Session \(./lenet7\)](#)

- Initial learning rate = 0.0015
 - o Learning rate after 15th epoch = 0.00015 (reduced 10% of original lr)
- **Model Trained and Validation accuracy is: 0.9673469387755103**

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.

Cell 18: Graph of Validation Accuracy with Epochs



Cell 19: EVALUATING given TEST dataset

- Restore session (./lenet7)
- Test Accuracy = 0.957

My final model results were:

- validation set accuracy of : 96.7%
- test set accuracy of : 95.7%

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?
 - Simple Lenet Architecture from previous Lessons , to check if datasets and models are working
- What were some problems with the initial architecture?
 - Accuracy of initial Architecture was around 87 to 88%
 - Due to small size of images (32x32), color mapping was good but not very good.
- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to over fitting or under fitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

- Added 1x1 convolution to improve color mapping. Validation Accuracy improved by 3 to 4%.
- Added Dropouts and Regularization to compensate for Overfittings. Tried range of biases [0.5, **0.05**, 0.005, 0.0005] and 0.05 worked best
- Added 3rd convolutional layer almost same as first two(from lenet) except Strides and kernel for max_pool kept to 1x1
- Which parameters were tuned? How were they adjusted and why?
 - Keep probability tuned to 0.8 for training (tried range from 0.5 to 0.93)
 - Tried range of biases [0.5, **0.05**, 0.005, 0.0005] and 0.05 worked best
 - Learning rate 0.0015 and 0.00015 (first 15 epochs and later 15 Epochs). Range of Learning rates experimented with was from 0.09 to 0.00009
- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?
 - Almost explained everything above
 - CNN: ConvNet structures make the unequivocal supposition that the information sources are pictures, which permits us to encode certain properties into the design. These then make the forward function more effective to execute and boundlessly lessen the measure of parameters in the system.
Good Article here: <http://cs231n.github.io/convolutional-networks/>
 - Dropout: It is an exceptionally productive method for performing model averaging with neural systems

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 10 German traffic signs that I found on the web:

Cell 20: def [process_newimage_file](#)(name):

- Read, Resize, Color BGR2RGB Images

Cell 21: Call function in Cell 20

- Load images into matrix 'newdata'
- Load respective signname indexes and appropriately format them

Cell 22: Visualization of these 10 images

Cell 23: def [pre_process_image](#)(X_in, Xb_out):

- pre-process : same function as was for training data preprocessing , But no concatenation (would have created 20 images : 10+10)

Cell 24: Visualization of preprocessed data

[0, 1, 2, 3, 4]



[5, 6, 7, 8, 9]



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

Cell 25: Restore session (./lenet7)
- Web-downloaded dataset Accuracy = 0.700

Here are the results of the prediction (from Cell 26 and 27):

Image	Prediction
Priority Road	Priority Road
30km/hr	30km/hr
Traffic Signals	Pedestrians (because of second white signboard below traffic sign in image)
Road work	Road work
100 km/h	30 km/h (Because Sign Board to Image ratio is less than 50%)
Wild Animals Crossing	Slippery Road (Because of Flipped image compared to X_train)
Keep right	Keep right
No Passing	No Passing
Stop	Stop
50 km/h	50 km/h

The model was able to correctly guess 7 of the 10 traffic signs, which gives an accuracy of 70%.

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)

Cell 26: Restore session (./lenet7)
 - Run logits, Predicts, and softmax for probabilities
 Cell 27:

