Traffic Sign Recognition

By ~Karanj

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the data set (Dataset)
- 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

---

1. project code.

You're reading it! and here is a link to my project code with execution and outputs

Data Set Summary & Exploration

1. Basic summary of the data set. Analysis is 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)

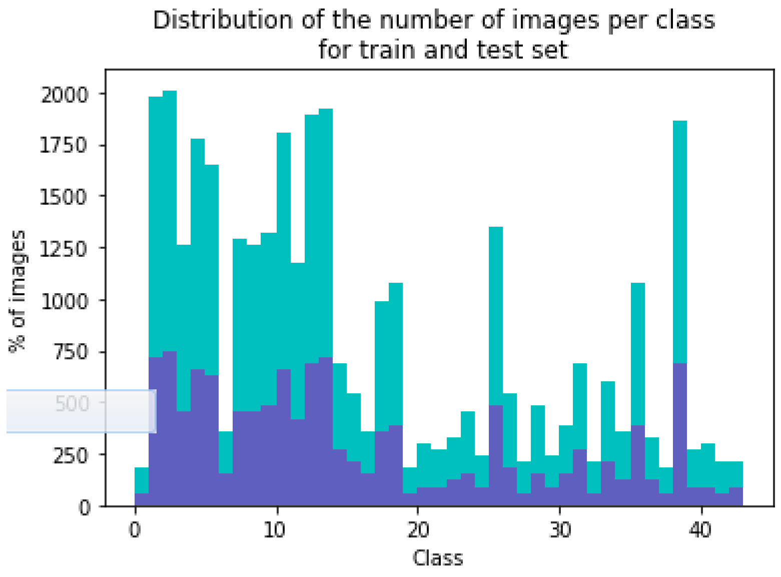
  Training Set: 34799 samples
  Validation Set: 4410 samples

Test Set: 12630 samples

Image data shape = (32, 32, 3)
Number of classes = 43

2. An exploratory visualization of the dataset.

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. Preprocessing the image data. 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
- Justification: Normalization tries to balance out brightness range between 0 to 255 so darker images shifts to brighter and brighter images shifts to darker so as to give image which may be viewed as under normal visual brightness. Augmentation provides little variation to the Normalized variation so as to help model to learn features of images under different lighting conditions.

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

- Rotation
- Translation
- Shift shear
- Call Cell 6 function
- Justification: Rotation, Translation, Shift shear provides modification to images so that to gather images which seems to be taken from a moving object (a car), where viewing angle, visual part of image to camera and tilt may not be in ideal position.

2. Set up training, validation and testing data. How much data was in each set. Generated additional data for training.

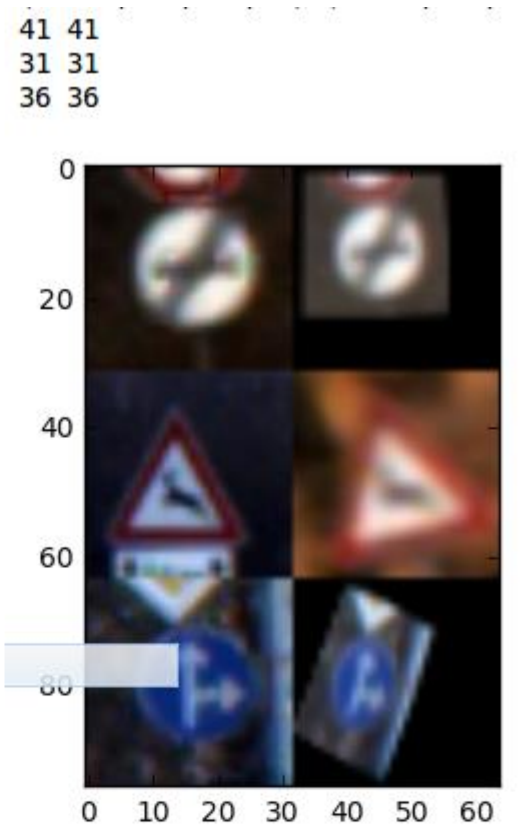
Cell 8: def preprocess_images(Xbtin_train, ybtin_train, Xbto_train, ybto_train):

- 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



```
41 41
31 31
36 36
```

3. Final model architecture, including model type, layers, layer sizes, connectivity, etc.

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. 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 : Adam Optimizer
- training_operation

Cell 16: Model evaluation

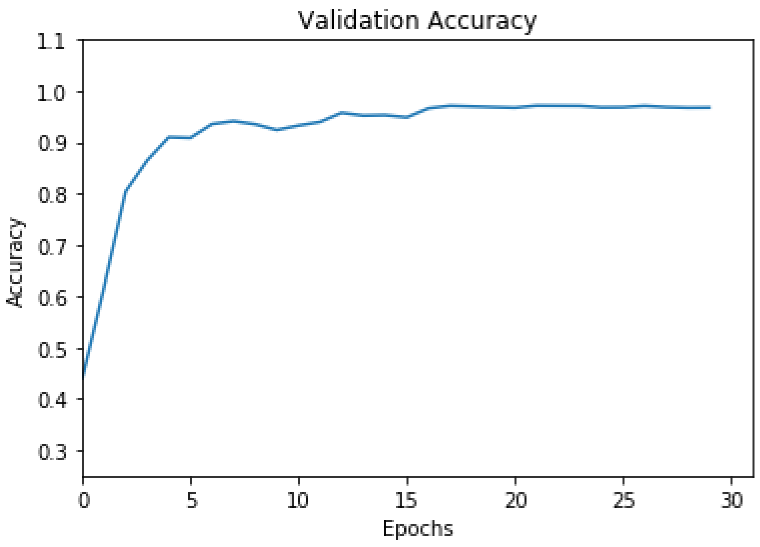
- correct_prediction
- accuracy_operation
- def evaluate(X_data, y_data):

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

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

5. The results on the training, validation and test sets.

Cell 18: Graph of Validation Accuracy with Epochs



Cell 19: EVALUATING given TEST dataset

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

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 bu 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.

Here are 10 German traffic signs that I found on the web, which resembles brightness, contrast and viewing angle as of training set data. After doing preprocessing the similarities between these images and training set data should increase as part of brightness normalization and Augmentation:

Cell 20: def process_newimage_file(name):
- Read, Resize, Color BGR2RGB Images

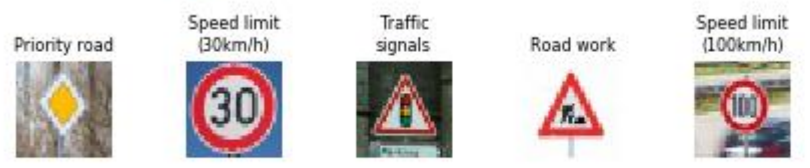
Cell 21: Call function in Cell 20
- Load images into matrix ‘newdata’
- Load respective signname indexes and appropriately format them

Cell 22: Visualization of these 10 images

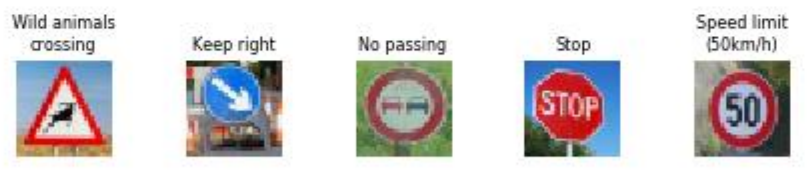
Cell 23: def pre_process_image(X_in, Xb_out):
- pre-process: same function as was for training data preprocessing , But no concatenation (would have created 20 images : 10+10)

Cell 24: Visualization of preprocessed data

```
[0, 1, 2, 3, 4]
```



```
[5, 6, 7, 8, 9]
```



2. Model's predictions on these new traffic signs.

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%. While the accuracy on test dataset was 95.7%.

Accuracy can be increased for downloaded images, if:

- images are more similar to 43 classless given in dataset
- classes in dataset could be increased (~= 200 to 500) to include many more traffic signs and from many countries.
- Same signs but different container shapes (triangle, squares, circles)
- Flipped signs ('Road work' sign can have image of person digging on right side and left side)

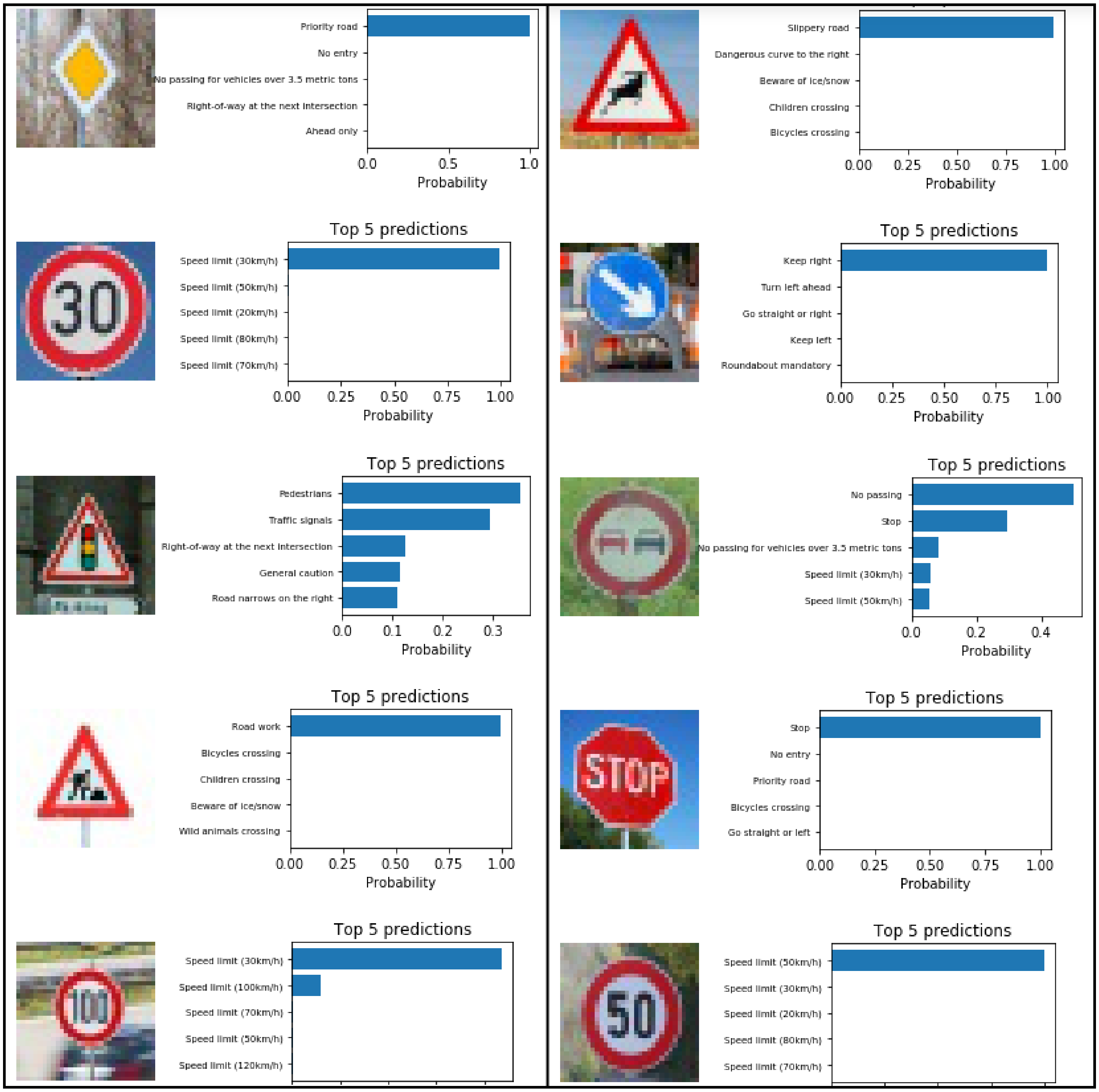
Certainty of model: If you combine the explanation of failing images in table above, 4 points I mentioned above to increase accuracy and sofmax probabilities on next page, it comes out that model developed in this project is good for local set of images (43 class - German signals), indeed should give accuracy of more than 94%. But for global perspective I would call it 'terrible' ☺ ☺, again because of 4 points mentioned above.

3. Softmax probabilities for each prediction. Top 5 softmax probabilities for each image along with the sign type of each probability.

Cell 26: Restore session (./lenet7)

- Run logits, Predicts, and softmax for probabilities

Cell 27:


Priority road
No entry
No passing for vehicles over 3.5 metric tons
Right-of-way at the next intersection
Ahead only
Probability
Slippery road
Dangerous curve to the right
Beware of ice/snow
Children crossing
Bicycles crossing
Probability
Top 5 predictions
Speed limit (30km/h)
Speed limit (50km/h)
Speed limit (20km/h)
Speed limit (80km/h)
Speed limit (70km/h)
Probability
Top 5 predictions
Keep right
Turn left ahead
Go straight or right
Keep left
Roundabout mandatory
Probability
Top 5 predictions
Pedestrians
Traffic signals
Right-of-way at the next intersection
General caution
Road narrows on the right
Probability
Top 5 predictions
No passing
Stop
No passing for vehicles over 3.5 metric tons
Speed limit (30km/h)
Speed limit (50km/h)
Probability
Top 5 predictions
Road work
Bicycles crossing
Children crossing
Beware of ice/snow
Wild animals crossing
Probability
Top 5 predictions
Stop
No entry
Priority road
Bicycles crossing
Go straight or left
Probability
Top 5 predictions
Speed limit (30km/h)
Speed limit (100km/h)
Speed limit (70km/h)
Speed limit (50km/h)
Speed limit (120km/h)
Top 5 predictions
Speed limit (50km/h)
Speed limit (30km/h)
Speed limit (20km/h)
Speed limit (80km/h)
Speed limit (70km/h)