Traffic Sign Classification Project

Data Augmentation by Image processing:
I have analyzed the training data provided by the Udacity for this project and noticed that it is very unbalanced. In order for the trained model to not have any bias towards the labels that have more examples than rest, I have augmented the training data by generating samples using image-processing techniques. This will also enable model to become more robust and be generalized. I have used 4000 samples for each of the traffic sign labels in the training data set.

- Horizontal Flip: flip the image horizontally for some image classes and skip it for others, as a flipped version will fall into other class
- Rotate: rotate an image with a random angle between (-45, 45)
- Translate: translate the image along X & Y axes by an offset in the range (0, IMAGE_SIZE/3)
- CLAHE equalization

Pre-processing training data:

- *Shuffle:* I have shuffled the training data during each epoch in order for model to learn features, independent of the order of training data.
- *Gray-scale:* Training data set has been converted to gray-scale and it will enable model to extract and learn the features in a more robust way. This might also reduce the noise related to varying colors in the image.

Model Architecture:
I have used a Convolutional Neural Network (CNN) to classify traffic signs. CNNs perform well for image classification by optimizing the model architecture. CNNs take advantage of translational invariance of images and capture features efficiently with less number of parameters compared to a regular neural network, which in-turn avoids the problem of over-fitting.

My model architecture uses 4 convolutional layers for feature extraction followed by 3 fully connected layers for classification. Features from all 4 convolutional layers have been used for classification purposes.

ELU:
For activation, I used Exponential Linear Unit (ELU) as it seemed to perform better than different variants of RELU. ELU avoids the issue of vanishing gradient and also enables in improved accuracy as well as faster convergence.

Dropout:
I have employed Dropout with a probability of 0.5 to avoid over-fitting. Dropout mechanism helps in making the model more generalized by dropping the links between layers randomly. In effect, it's like training over an ensemble of networks, for each iteration and averaging out the results.

L2 Regularization:
I have also employed L2 regularization with a lambda of 0.0001, to avoid over-fitting. For applying this regularization, I used convolutional layers' as well as fully connected layers' weights.

Softmax Cross Entropy & Adam Optimizer:
I have used softmax cross entropy and L2 regularization to compute the cost function which needs to be minimized using Adam Optimizer. I have used a learning rate of 0.0005, as this seems to be optimal between accuracy and speed of convergence.

Model Training Parameters:
Epochs = 200
Batch size = 128

Metrics:
Training accuracy: ~94%
Test accuracy: ~92%

References:

1. http://www.pyimagesearch.com/2017/01/02/rotate-images-correctly-with-opencv-and-python/
2. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histograms/py_histogram_equalization/py_histogram_equalization.html
3. http://stackoverflow.com/questions/24341114/simple-illumination-correction-in-images-opencv-c
4. https://en.wikipedia.org/wiki/Adaptive_histogram_equalization
5. http://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html
6. http://navoshta.com/traffic-signs-classification/
7. https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization
8. http://image-net.org/challenges/posters/JKU_EN_RGB_Schwarz_poster.pdf
9. http://www.picalike.com/blog/2015/11/28/relu-was-yesterday-tomorrow-comes-elu/
10. https://www.quora.com/How-does-ELU-activation-function-help-convergence-and-whats-its-advantages-over-ReLU-or-sigmoid-or-tanh-function