Assignment 3

THE GERMAN TRAFFIC SIGN DETECTION BENCHMARK

Roshan Kumar(rk3110) | Computer Vision | 12/8/2016

Abstract

In this assignment, we apply different versions of Convolutional Networks and other techniques to the problem of Traffic Sign classification as part of Kaggle Competition. Broadly speaking, I have applied the following techniques as part of my experimentation:

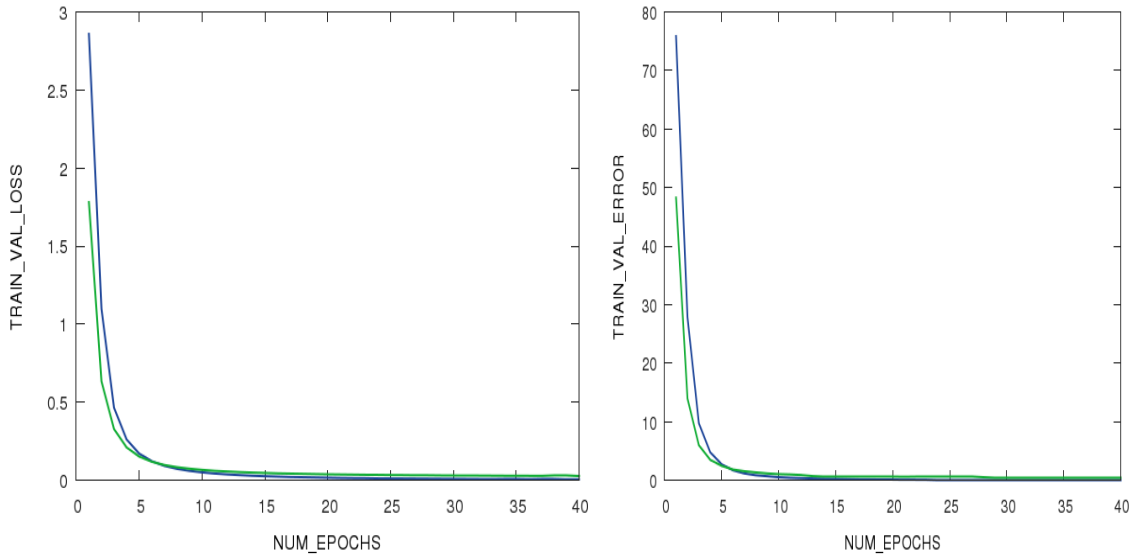
- Run the baseline model provided
- Perform some preprocessing such as scaling and normalization on the dataset and run the given baseline on it
- Augment the dataset by adding a jittered dataset that enhances robustness of model to various transformations and run the baseline model.
- Crop image around Regions of Interest, available as part of dataset and see the effect on accuracy.
- Replace the baseline model with VGGNet and run it on preprocessed and augmented dataset.

Baseline Model

Trained the baseline model for 300 epochs.

Accuracy : 0.94885

Model Convergence plots for only 40 epochs due to server limitations:



Base model with preprocessing and data Augmentation:

Performed following preprocessing as indicated in the paper at http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf:

1. Dataset sample sizes vary from 15*15 to 250*250. To simplify the processing of images, we down sample images to 32*32.
2. Convert RGB to YUV space.
3. Pre-process all 3 channels with global and local contrast normalization. Global normalization centers image around its mean value whereas local normalization emphasizes edges.

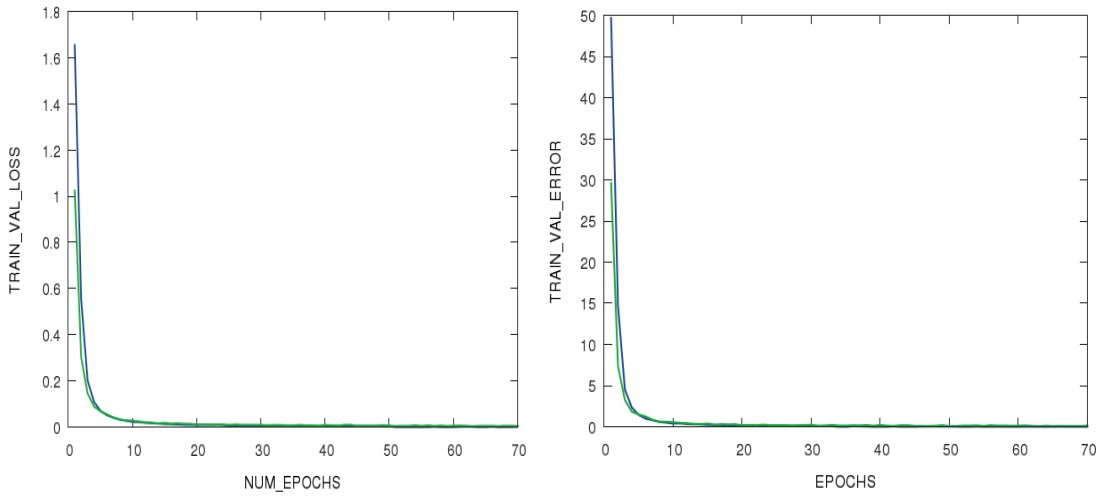
Data Augmentation:

Additionally, I built a jittered dataset by adding a couple of transformed versions of the original training set yielding a dataset three time the given dataset. These were dataset obtained translating and rotating the original image. It increases robustness of model to translation and rotation. Consequently, this greatly increased the accuracy of the model.

Trained the model for 80 Epochs and obtained the following accuracy and convergence plots.

Accuracy: 0.97862

Convergence Plots:



Additionally, I performed cropping of both train and test images around the given Regions Of Interest which additionally increased the accuracy as follows:

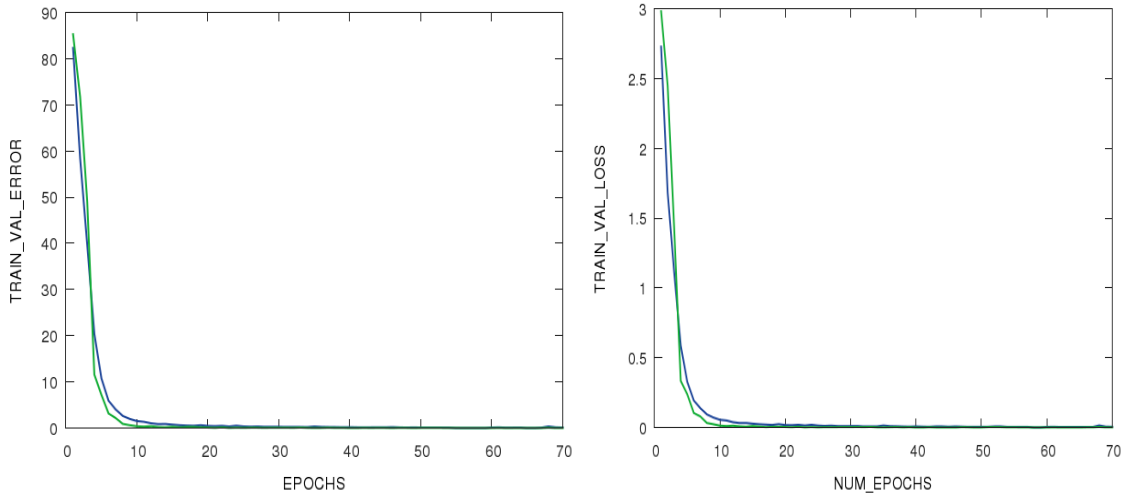
Accuracy: 0.98116

Best Performing Model

Model description: Replace base model with a VGG in the above model using exactly same preprocessing and adding a jittered dataset as above. Training the model for 70 epochs yields following accuracy and convergence plots:

Accuracy : 0.98242

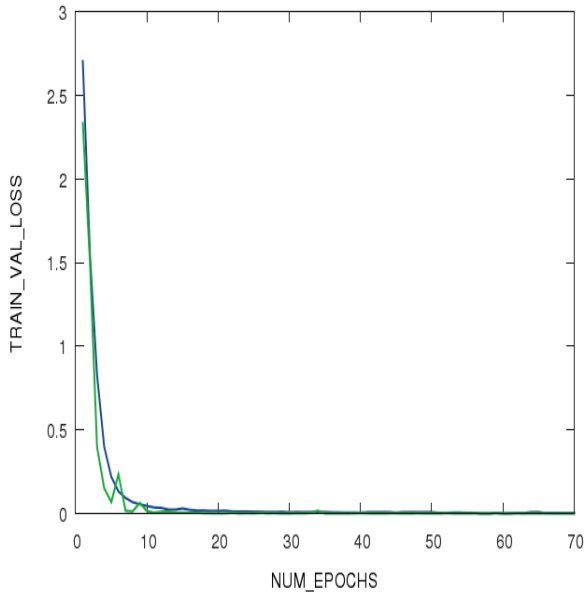
Convergence plots:

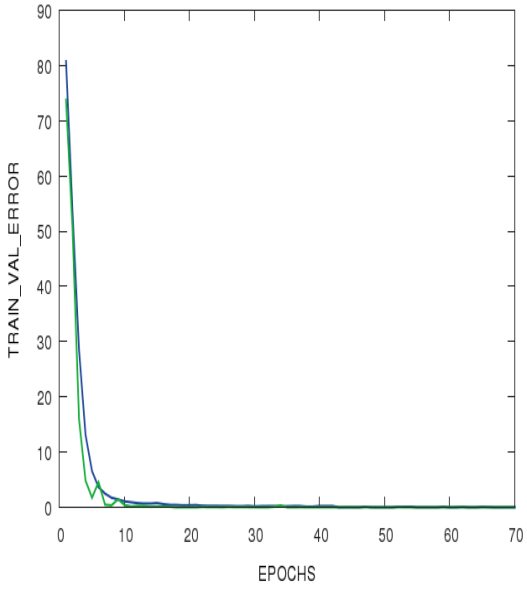


Additionally, cropping the both test and train image around region of interest(ROI) as done above yields best accuracy as reported below:

Accuracy : 0.98987

Convergence Plots:





Floating Point Operation Calculation:

In the VGG net model which gave best accuracy, there are 13 convolutional layers; #FLOP for a single convolutional layer can be computed using the following formula: #inputPlane*#outputPlane*width*height*kernelWidth*kernelHeight.

Conv layer 1: 3*64*32*32*3*3 = 1769472

Conv Layer 2: 64*64*32*32*3*3 = 37748736

Conv Layer 3: 64*128*16*16*3*3 = 18874368

Conv Layer 4: 128*128*16*16*3*3 = 37748736

Conv Layer 5: 128*256*8*8*3*3 = 18874368

Conv Layer 6: 256*256*8*8*3*3 = 37748736

Conv Layer: 256*256*8*8*3*3 = 37748736

Conv Layer 8: 256*512*4*4*3*3 = 18874368

Conv Layer 9: 512*512*4*4*3*3 = 37748736

Conv Layer 10: 512*512*4*4*3*3 = 37748736

Conv Layer 11: 512*512*2*2*3*3 = 9437184

Conv Layer 12: 512*512*2*2*3*3 = 9437184

Conv Layer 13: 512*512*2*2*3*3 = 9437184

Total number of FLOP can be obtained by summing up the above.

Conclusion:

From the experiments above we can see the effect image preprocessing has on the performance of model. Also, how model can be made more robust to transformation by training it on various transformation of given dataset. The accuracy can be further increased by further augmenting the dataset with other transformations of input image such as horizontal flip, vertical flip, center crop etc.