PROJECT: Traffic Sign Recognition (Documentation of Results)

Abhishek Jagtap – 17-02-21

Contents:

Introduction:

This report sheds light on how traffic signs classification problem can be tackled with DNN implemented using Pytorch by documenting results of model performance. The models are trained on German Traffic Sign Dataset (German Traffic Sign Recognition Benchmark) and reaches 99.23% accuracy using spatial transformer network. The pipeline is included with experimenting and fine-tuning methods to improve forecasting accuracy on test data.

Dataset Summary and Exploration:

The GTSRB dataset provided by (German Traffic Sign Recognition Benchmark) consist of multi-class classification problem, images are spread across 43 different classes containing a total of 32,799 train images and 12,630 test images. The dataset is pickled, and the images are resized to 32×32. The below Fig [1] and Fig [2], shows the number of counts of each class and a batch of data.

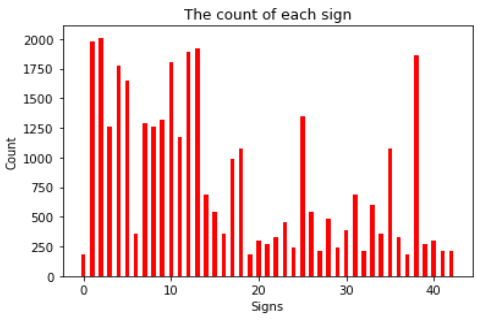

Fig [1] - Class distribution of Dataset

Fig [2] - Batch of Data

Experiment-1:

In this section, the pipeline is run with default configurations, the pipeline consists of six different models including Spatial Transformer network. The results obtained by each model and approach is documented and compared sequentially. For visualizing the results and performance of our network, TensorBoard tool is used. All the required plots and graphs documented in real time are stored in the attached zip file under runs folder.

Model-1: BaselineNet Model

This model consists of two convolutional layer which takes the RGB images as inputs and is propagated through Max-pooling layers and fully connected layers. Loss function, hyper-parameters such as batch size, epoch and learning rate are configured default and Test accuracy of 85.18% is achieved. The model graph and corresponding observations are shown in the figure [3] and [4] respectively.

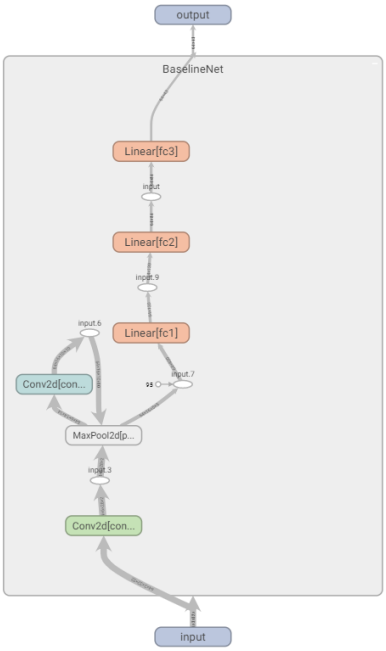

Fig [3] -BaselineNet Model Graph

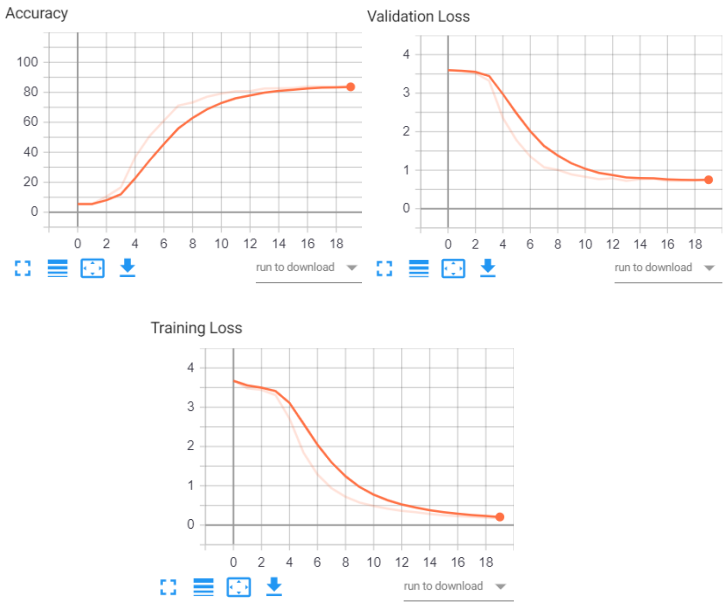

Fig [4] -Observations of BaselineNet Model

Model-2: BaselineNet Model with Data Pre-processing

CLAHE (Contrast-Limited Adaptive Histogram Equalization) is used to remove noise and intensity inhomogeneities in the image while retaining the natural gray level variations, the resulting data is then fed as input to BaselineNet model. The resulting accuracy of the model is 86.595%. The preprocessed data is shown in fig [5] and accuracy is plotted in fig [6].

Model-3: Baseline Model with Data Pre-processing by Handling Imbalanced Dataset

From Fig [1] we can concur that there is class imbalance in the given data, thus more care is given to minority classes in this step, so we extend our training data by flipping the number

of counts of each sign. The distribution of dataset after performing this trick is shown in the Fig [7]. The resulting input is then propagated to our Baseline model which in turn yields an accuracy of 88.116% on testing data, which is slightly more compared to model 2.

Fig [5] - Pre-Processed Data

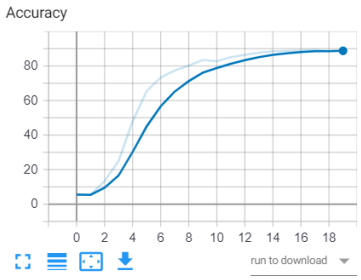

Fig [6] - Accuracy (Model-2)

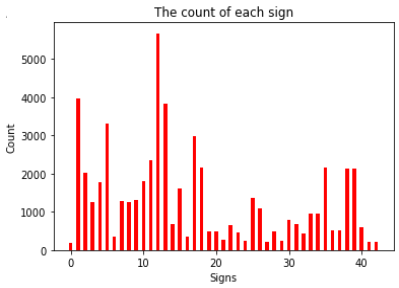

Fig [7]- Dataset distribution after flipping

Model-4: Baseline Model with Augmented Data

In this approach, the diversity of the data available is increased by performing geometrical transformation such as rotational, translational, shear and scaling the same sign images. Training the model with this augmented data results in a good results with validation accuracy of 95.488% and Test accuracy of 93.967%. The below Fig [8] show the performance of Model-4 compared to previous approaches.

Fig [8] - Documentation of results w.r.t each model

Model-5: TrafficSignNet Model

In this model, regularization techniques are incorporated such as batch normalization, dropout and early stopping to prevent overfitting of the model and Adam optimizer is used instead of SGD and hyper parameters such as learning rate, batch size and epochs are configured to 0.0001, 64 and 100, respectively. The results obtained by this model is exceptional when compared to all previous models, the training terminated for early stopping at epoch 57, the trained model generated accuracy of 99.058% on testing dataset. Simultaneously, to understand more about the convolution layers in TrafficSignNet model, a graph is visualized and can be seen in Fig [9].

Model-6: TrafficSignNet with Spatial Transformer Network Model

CNN are still inefficient in learning Spatial invariance of the input data, thus STN is introduced which allows spatial manipulation of the data within the network. The model could achieve an accuracy of 99.232% on 12,630 new testing images, the above observation Fig [10] shows how fast STN model could generalize and provide state of the art performance on GTSRB dataset when compared to TrafficSignNet model.

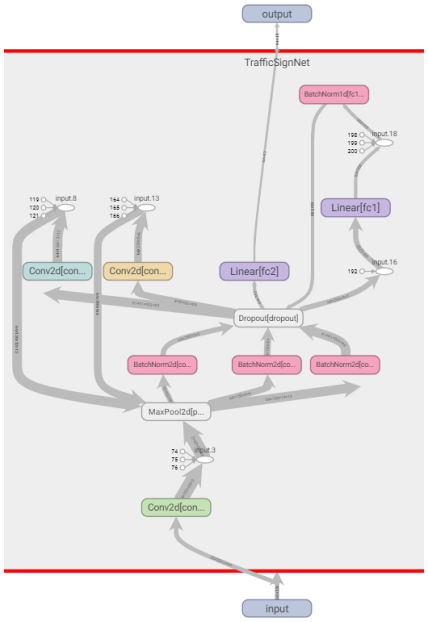

Fig [9] - TrafficSignNet Graph

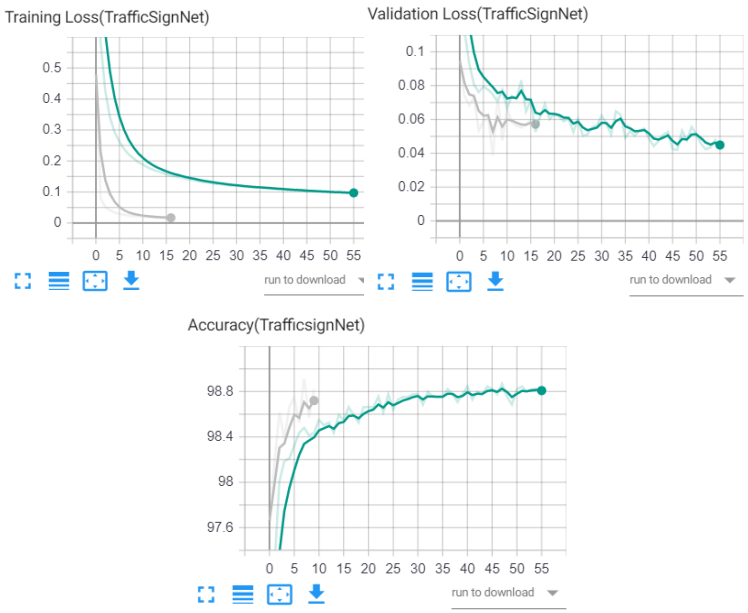

Fig [10] - Performance of TrafficSignNet compared to STN Model

Experiment 2:

In this section, Spatial transformer network is disabled, and the re-run of the training procedure with the same pipeline is conducted and corresponding results were captured in real time in tensorboard for comparing the performance of the models and can be seen in Fig [11]. The models are trained on same hyperparameters as in experiment 1, a decrease of 0.055% in testing accuracy is observed when compared to model incorporated with STN.

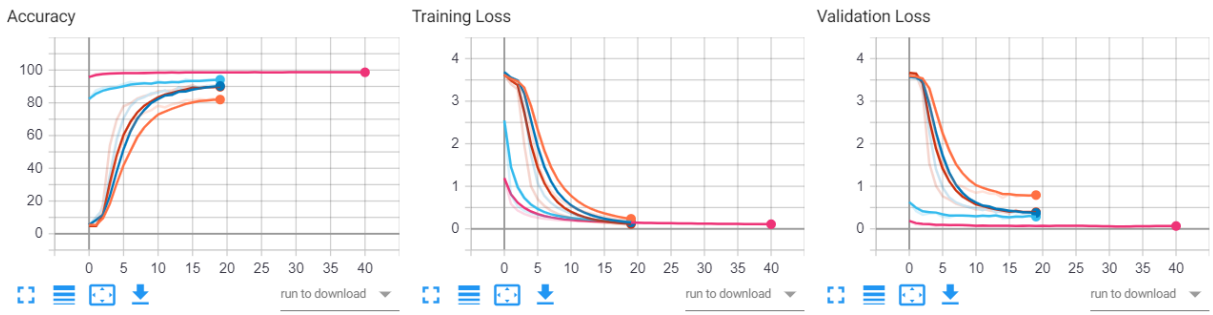

Fig [11] – Documented Results after Training [Experiment 2]

Experiment 3:

In this section, we visualize the activation of the third convolutional layer for 10 different images and compare them with our models in experiment 1 and experiment 2. The Fig [12] shows one such input image fed to TrafficSignNet Model and its corresponding activation of third convolution layer is visualized in 16 two dimensional images as shown in Fig [13].

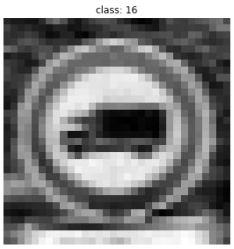

Fig [12] -Input image (class 16)

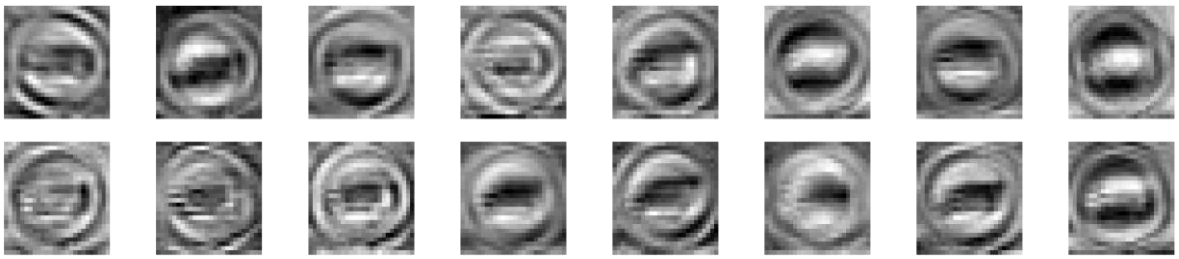

Fig [13] - Activation layer of Conv3 layer (class 16)

In STN model, the below Fig [14] is fed as input and the corresponding activation of the convolution layer is visualized in similar fashion and can be observed in Fig [15], by visualizing this intermediate output we get the idea of what features of the input are detected or preserved.

Fig [14] -Input Image (STN model)

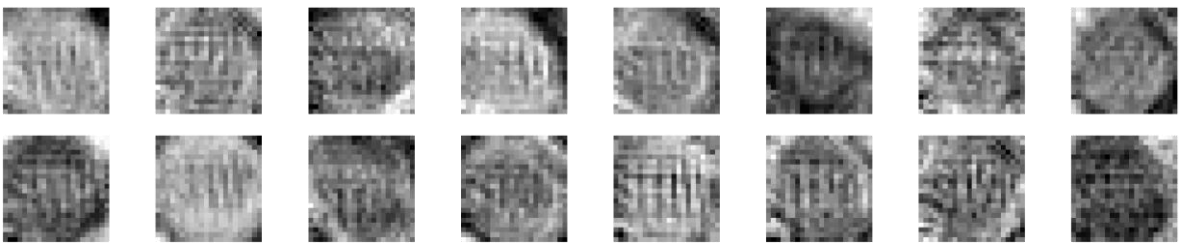

Fig [15] - Activation of second convolutional layer (STN model)

Conclusion:

After training the model through various pre-processing stages and augmenting the data a state-of-the-art performance is achieved with an accuracy of 99.232% on new testing data.

For visualization, tensorboard tool is used to plot various graphs and charts but one challenge faced is I could not upload the tensorboard into an accessible weblink due to some issue in my laptop, so a tensorflow events file is attached in zip file containing summary of all the runs conducted.

In experiment 3, visualizing activation of layer conv3 was not achieved efficiently on STN Model, and I had hard time getting gradients of the layer for the same images. I realized at the end a different approach such as forward hook for visualizing activations would have been more feasible.