Assignment 3: Kaggle Competition GTSRB
Computer Vision, NYU

Amanpreet Singh

December 2017

1 Summary

I know that this report needs to be brief but to clearly present my ideas, I will have explain them in detail. That is why, I have included this summary for your convenience. Following points explain the models and data augmentation used:

1. Tackled class imbalance with flipping of original images and use transformation (affine and projective) with rotation and brightness changes to generate new data
2. Used Spatial Transformer Network modules with modified IDSIA (batch normalization, dropout and other new layers added) network to tackle distortions in data and achieve an accuracy of 99.4% (with augmentated data) and 98.9% (without augmentations).
3. Other models used: Multi Column Deep Neural Network (MCDNN, IDSIA) alone which achieve an accuracy around 98.6% (without using any augmented data) and three layer Convolutional Neural Network with Dropout and Batch Normalization layers which achieves around 97.7% (without using any augmented data).

2 Introduction

Assignment requires us to implement models which will perform on German Traffic Signs Recognition Benchmark dataset [1]which is publically available. We have been provided with starter code, train images and labels and test images. We generate a CSV file to upload on Kaggle which is scored based on error metric.

3 Issues with the data

There are numerous issues with dataset which represent the real world problems faced in actual traffic sign recognition. We enlist the issues with dataset in the following points:

1. **Class Imbalance**: There is a serious class imbalance in the dataset, with some of the classes having 2000 samples (class 1, 30km/h sign) and some of them having only 200 samples (class 0, 20km/h sign), which is a representation of the real world distribution of these classes as 30km/h sign is likely to occur more than 20km/h sign which occurs only in school areas. This class imbalance biases the model to predict classes with higher number of samples more frequently than the ones with less number of samples to achieve better accuracy. So, our model needs to be robust towards class imbalance.

2. **Deformations**: A lot of the images are taken in bad lightning and are distorted with various kind of distortions ranging from translation, rotation, brightness and lightning. It is difficult for even human to understand and classify some of these signs which are in total darkness. These distortions add an extra layer of complexity in correctly classifying these images. Our model will need to be robust towards these distortions in data. As per [2], we use only one channel Y from YCbCr color space.

3. **Small dataset**: Even though we have 64k train images, they are not quite enough to perform well in all general case scenarios which includes all kind of distortions. More data also helps with overfitting. With current amount of data it is very easy to overfit and perform bad on test set. More data will also help with tackling distortions. More data in general is always good.

4 Solutions adapted for issues

We enumerate the solution that we adopted for each of the issues above:

1. **Class Imbalance**: We first tried to use our custom sampler which samples all of the classes with equal amount and passed that to data loader of pytorch. Issue with this approach is that we need to set shuffle as false in data loader as sampler and shuffle are mutually exclusive option in data loader. This change did improve our model's accuracy but not much. To remove class imbalance, we decided to follow data augmentation techniques like scaling, rotation, shearing and projection transformation to achieve same number of classes for each class. We use scikit-image library to perform all kinds of augmentations [3].(For training, we augment to create 10k samples for each image)

2. **Deformations**: These are best handled by creating new samples which contain all type of deformation. Using color jitter to introduce brightness and saturation changes with rotation/affine and projective transformations helped us to create new samples which contain these various kind of deformations which can happen at test time. In general, we try to generate a sample for any kind of distortion that can happen to a sample for our training set.

3. **Small dataset**: We use various similarities between classes to increase are natural set of images. For example, turn left sign when flipped horizontally becomes turn right; 30km/h sign can be flipped vertically to create new

image of same class; traffic light sign can be flipped horizontally to create new image of same class; end of speed signs can flipped both vertically and horizontally to create new image. Thus, by flipping we get our natural set of new images. We further, augment with different intensities to create balanced datasets.

We first train our model on extended dataset, which contains flipped and original version of images, to let the model get a feel of how data actually looks like as original distribution should be a good estimator of test dataset. Then, we train our model on balanced dataset which contains equal number of samples for all classes generated via augmentations. We also use 25% of the training set we are using at a particular time as a validation set as original validation set that came with the starter isn't a good representative of overall dataset and its size is quite small.

Using all these augmentations with our models, we are able to achieve best test accuracy of % at the time of this writing.

5 Models Implemented

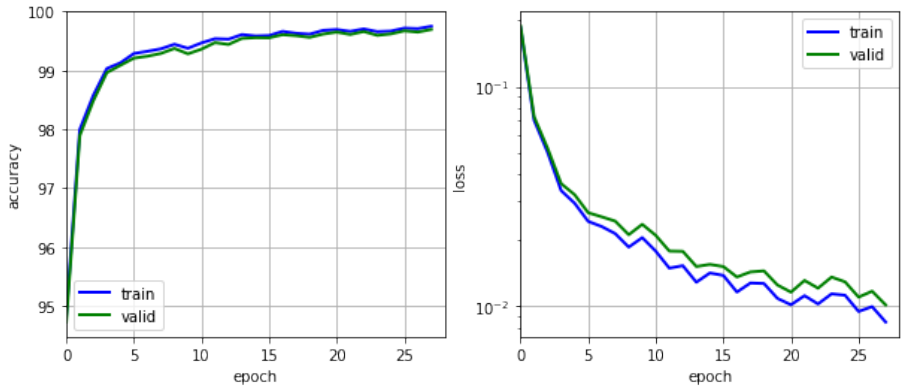
Following were implemented and tested on GTRSB dataset as a part of this assignment by us:

1. Spatial Transformer Network: Currently, state of the art results are held by a model [4] which uses Spatial Transformer Networks [5] and uses both local and global features through spatial normalization (both locally and globally). STNs help in mitigating deformations in images such as rotation, translation and scaling. Inspired by this work, we also implement a similar model, which uses spatial transformer networks and a modified IDSIA network, and train it with heavily augmented data to reduce class imbalance. We add multiple dropout and batch normalization due reduce data invariance and overfitting. This implementation achieves around 99.4% test accuracy. Checkout *network.py* file in the repository for model topology. See README of Github repository for more details.

2. Multi-Column Deep Neural Network [6]: We also implemented the original IDSIA MCDNN model with extra batch normalization layer and number of other modifications. This model performs quite well on the original data without augmentations and is able to achieve around 98.6% accuracy for us. If you use our code without passing *st* flag it will use this modified MCDNN to train the model.

3. 3 Layer Convolutional Neural Network: We also trained a simple 3 layer convolutional network with 2 fully connected layers and high number of channels (150, 200 and 300). We add dropout and batch normalization layers to this CNN and train it on original data. We achieve a test accuracy of around 97.7% with this simple model.

4. 2 Layer original model: This CNN was provided with the starter code. We train it on original data to achieve best test accuracy of around 92%. Adding a single layer with batch normalizations and convolving to more channels greatly improves the model.



Training/Validation Accuracy and Loss vs Number of epochs

All of the above models were trained with early stopping (patience of 10), adam optimizer (lr=0.0001), cross entropy loss and dropout of 0.5. Validation cross entropy loss was minimized to find the best model, as we want our model to be confident of its prediction so that it can do well on generalized test set. As we use huge amount of augmented data, we easily reach validation accuracy (25% set) of 93% after first epoch.

6 Results

We see that using augmented data to introduce all kinds of distortions to our model, removing class imbalance and using spatial transformer network which are robust towards deformations with a deep convolution neural networks helps us achieve test accuracy. This even strengthens the fact that in GTSRB traffic signs dataset, it is critical to tackle both deformations and imbalance to achieve a good test accuracy. Spatial Transformer Networks are a good building blocks in classification tasks where there are a lot of distortions. The code is available for use on Github [1] and contains a readme explaining how to run it. Current standings for the competition can be seen on Kaggle [2]. It would also be nice to explore if accuracy can further be improved by using skip connections [2]. As we achieve a 99.9% accuracy on all kinds of validation sets tried by us, it would interesting to see on which samples our models fails in test set once we have access to the labels.

References

[1] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1453–1460. IEEE, 2011.

[2] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2809–2813. IEEE, 2011.

[1] https://github.com/apsdehal/traffic-signs-recognition
[2] https://www.kaggle.com/c/nyu-cv-fall-2017/leaderboard

[3] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[4] Mrinal Haloi. Traffic sign classification using deep inception based convolutional networks. *CoRR*, abs/1511.02992, 2015.

[5] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[6] Dan CireşAn, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.