Udacity
Self Driving Car Engineer Nanodegree

Traffic Sign Classification

Jaganadh Gopinadhan

May 3, 2017

1 Introduction

Over the past decade, autonomous vehicles generated interest within academia and industry. Many successful industrial and academic initiatives in R&D in this area are in progress across the globe. The first two elements in developing autonomous vehicles is a) Localization and Mapping and b) Scene Understanding. Traffic Sign Detection and Classification is one of the most significant problems in this segment. Detection and classification of traffic sign from real-time video feed are challenging due to some issues. The key challenges are occlusion, color variation, rotation, skew, fast movement of the camera and resulting blur and environment facts like wind or dust, etc.. An efficient traffic sign classification system has to be capable of handling such conditions.

Traffic signs are part of the visual language of traffic flow controls and driving. Recognition and classification of the sign is a granted ability for human drivers. To a computer it a challenging problem falls under the category of image processing and pattern understanding. Accurate detection and localization of one or more traffic signs from a vehicle camera feed are referred as traffic sign detection. The result of this detection is used for sign classification. The task is a classic example of supervised Machine Learning problem. Face recognition, digit recognition and object recognition are some of the closely related tasks in image processing and Machine Learning. Our task is to develop a traffic sign classification model based on Deep Learning algorithms.

2 Data

The Dataset used for this experiment is from the German Traffic Sign Recognition Benchmark (GTSRB), provided by Udacity. The data was supplied as Python.3x compatible pickle files. Training, testing and validation set are provided as separate files. Each pickle file is a Python dictionary structure having the keys 'labels', 'coords', 'sizes', 'features'. The 'features' contain the image pixel data, 'coords' contains localization information, 'size' contains image size and the 'labels' contains a numeric value representing the traffic sign represented in the image. The train, test and valid data consist 34799, 12630, 4410 samples respectively.

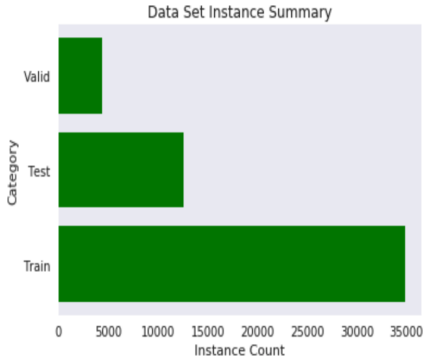


Figure 1: Data Distribution

There are 43 traffic signs in this image data sets, and it translates to 43 labels in the data sets.

Figure 2: Sample Images from 43 Classes

The size of the image varies from 20 to 250 (Image size distribution is presented in the Appendix). The dimensions of the image are 32x32x3, and all the images confine to the same in training test and validation data. An analysis of the class

distribution in training data suggests that it is a highly imbalanced dataset. The well-represented class consists samples in the range of 1000 to 2500 and the under-represented classed consists instances in the range of 120 to 850.

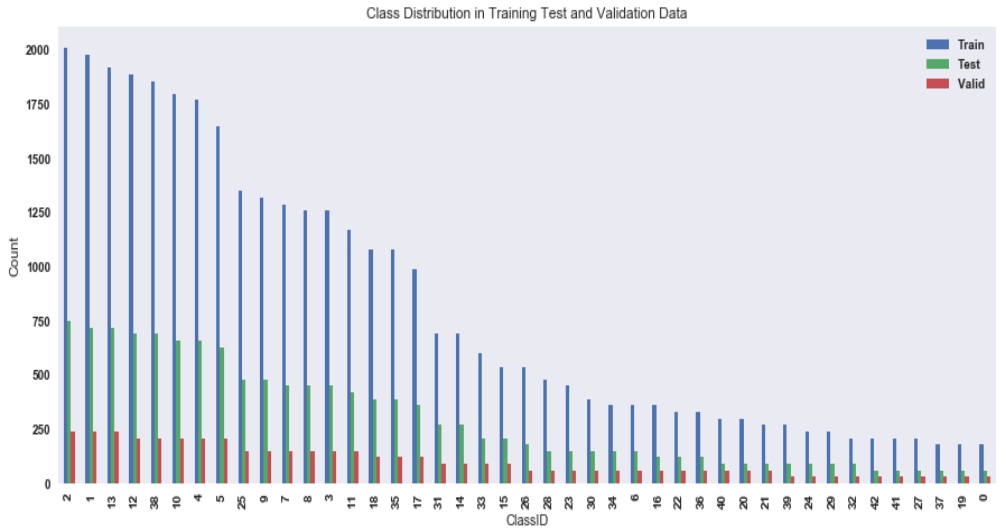


Figure 3: Class Distribution in Training Test and Validation Data

The images are taken in varying light conditions, perspectives, and conclusions. The natural decay of signs due to age and the climatic condition is also present in the data.

3 Preprocessing

Before the starting, the model building preprocessing steps were performed in the image data. The preprocessing steps carried out in the data are, a) data augmentation b) image preprocessing.

3.1 Data Augmentation

The training data is highly imbalanced in nature. There are two practical solutions to overcome this scenario. One way to get around the lack of data is to augment the data, and the other is under-sample over-represented categories and create an equal distribution. Smart approaches such as programmatic data augmentation can be used to increase the size of training data. This will help in preventing the over fitting to a limit (Over fitting can happen due to other reasons too). Depending upon the type of the data there are many techniques available for oversampling the

minority data. Imputation, adding noise to the data and transforming the data are some of the common techniques in image data augmentation. In this experiment, we have used the same approach for augmenting the data.

Two augmentation steps were performed in the training data. The first one was facilitated by Keras library. In this step image rotation, horizontal shift, vertical shifts, shear intensity, random zooming and horizontal flipping is performed. The settings used for the augmentation is:

- featurewise_center: Set input mean to 0 over the dataset, feature-wise.
- rotation_range: Degree range for random rotations. The rotation range was specified as 17 for the augmentation.
- width_shift_range: Range for random horizontal shifts. Specified value 0.1
- height_shift_range: Range for random vertical shifts. Specified value 0.1
- shear_range: Shear Intensity (Shear angle in the counter-clockwise direction as radians). Specified Value: 0.3
- zoom_range: Range for random zoom. Specified Value: 0.15
- horizontal_flip: Randomly flip inputs horizontally. Not performed

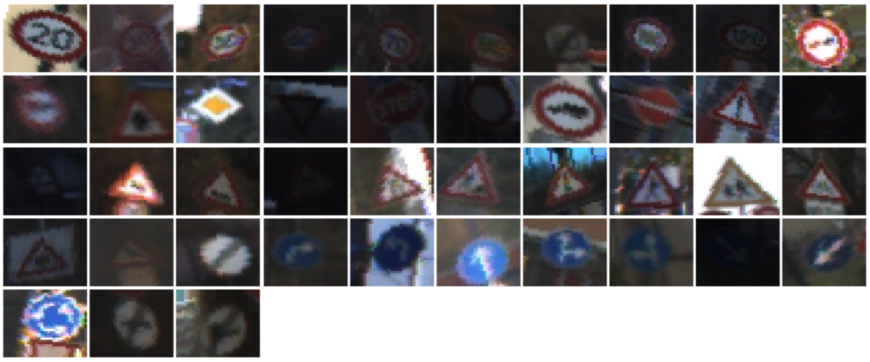


Figure 4: Sample Images - Augmentation

The second argumentation step is adding motion blur to the images. It is a noise adding technique. Motion blur is the apparent streaking of rapidly moving objects in a still image. It is a way to simulate a fast moving scene. It helps the traffic sign classifier to almost accurately classify a localized sign from blur image to the proper category.

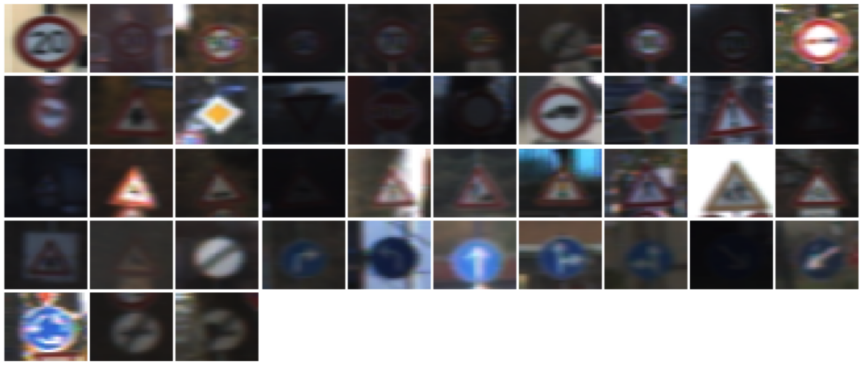


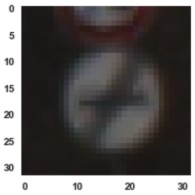
Figure 5: Sample Images - Motion Blur

3.2 Image Preprocessing

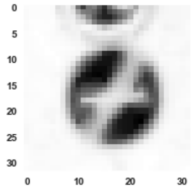
In this experiment, we preprocessed the images before it is being presented to the algorithm for model building. The preprocessing steps adopted in the projects are e converting the images to gray scale and normalizing the exposure. The gray scale conversion was done with RGB to YUV color transformation and then extracted the gray scale image. The YUV model defines a color space regarding luminance (Y) and chrominance (UV), luminance regards the brightness while the chrominance is the color component. The YUV color space can encode color images and videos such that any errors are more efficiently masked from the human perception. The key advantage of conversion to gray scale is, it simplifies the algorithm and reduces the complexity. In computer vision based applications, color to gray scale conversion helps to preserve the salient features of a color image, such as brightness, contrast, and structure. By using the YUV conversion, we are accounting the human perception model and then extracts the gray scale image. In this way, the salient features are preserved in the resulted image.

Normalization refers to normalizing the data dimensions so that they are of approximately the same scale. There are many normalization methods used in image processing such as divide each dimension by its standard deviation, normalizes each dimension so that the min and max along the dimension are -1 and one respectively. In images, the relative scales of pixel values will be approximately equal, and the general normalization techniques are not applied. But as a rule of thumb when we are playing with Deep Neural Nets it is critical to zero centers the data. In our experiment, we used Contrast Limited Adaptive Histogram Equalization (CLAHE) technique for image normalization. CLAHE is an algorithm for local contrast enhancement. It uses histogram computer over different regions of the same image. The key advantage of applying this technique is, it can enhance local regions even in regions that are darker or lighter than most of the images.

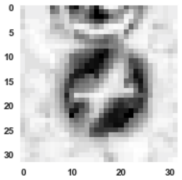
Figure 6 presents an example of YUV to grayscale image extraction and CLAHE



(a) Original Image



(b) Grey Scale Image Extracted from YUV



(c) After CLAHE

Figure 6: Gray Scaling and Normalization Steps Sample

technique. If we closely observe the image, it is evident that the visual semantics of the traffic sign is more visible in the gray scale image. After the CLAHE normalization, the features became more clear and we can see that the half circle pattern in the top of the image is more clear after the CLHE process. The combination of YUV and CLAHE in preprocessing helping us in grayscale the image as well preserve and enhance the salient featured of the image. From this preprocessed image it will be quite easy for a perfect convolution network to learn the minute details of each category of the traffic sign.

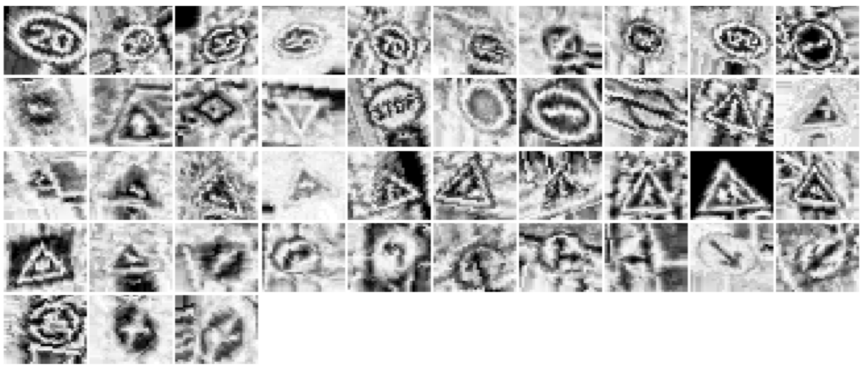


Figure 7: Sample Preprocessed Images

4 Model Architecture

As part of this exercise, we created four model. The first model is the baseline model and respective model as enhancements in the base model to improve the performance. The model was developed using Keras 2.0 with Tensorflow back-end (GPU). We used Convolution Neural Networks (ConvNet) algorithm for building the model. The details of each ConvNet architecture is discussed in detail.

4.1 Baseline Model

The baseline model architecture was adopted from a masters thesis from Chalmers University of Technology. The Thesis was written by Linnea Claesson and Bjorn Hansson. The title of the thesis is Deep Learning Methods and Applications Classification of Traffic Signs and Detection of Alzheimer's Disease from Images (http://studentarbeten.chalmers.se deep-learning-methods- and-applications-classification-of-traffic-signs-and-detection-of-alzheimers-d) . The model used in the thesis consists two convolution layers and two fully connected layer were the second fully connected layer is the output layer. To build our baseline model we introduced one more convolutions layer in the architecture. The network architecture of the model is given here.

Layer	Data	Type	Filters	Kernal	Padding	Strides	Activation	Pooling	Pool Size	Pool Stride	Dropout
Input	Image 32x32	Input Layer									
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.2
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.3
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	1x1	1x1	0.4
Fully Connected			600				RELU				0.6
Fully Connected							SOFTMAX				0.7

Table 1: Baseline Model Architecture

A Complete network flow diagram is provided in the appendix for reference.

We used the ADAM optimizer with learning rate 0.001 and decay 0.0. The loss function used was categorical cross entropy, and the evaluation metric for training is set to accuracy. The number of epoch was set to 10 as this is going to be the baseline model.However, we used an early stopping strategy to stop the training in the event of early convergence. The early stopping function will check the validation loss to determine the point when the training has to be stopped.

The training was stopped after seventh epoch with 0.92 accuracy in validation. The sixth and fifth epoch were reported 0.92 and 0.91 accuracy respectively. The accuracy of test data was 0.91 only.

Model	Validation Accuracy	Test Accuracy	Benchmark Accuracy
Baseline	0.92	0.91	0.93

Table 2: Accuracy

Model	Validation Accuracy	Test Accuracy	Benchmark Accuracy
Baseline Modified - I	0.0068	0.00475	0.93

Table 3: Accuracy

The model performance was below the benchmark score of 0.93. To beat the benchmark, we decided to alter the architecture and test it.

4.2 Enhanced Model - I

We added one more convolution layer to the baseline architecture to check if it is going to improve the performance. The same training setting used for Baseline Model is used for this model too. The only modification here is the number of epochs increased to 20. The model got converged at 6th epoch with a validation accuracy of 0.0068, which is not a good score. The accuracy of the test data is just 0.00475.

We discarded this model and altered the base architecture again. The model architecture is given below.

Layer	Data	Type	Filters	Kernal	Padding	Strides	Activation	Pooling	Pool Size	Pool Stride	Dropout
Input	Image 32x32	Input Layer									
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.2
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.3
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	1x1	1x1	0.4
Convolution		Conv2D	16	5x5	valid	1x1	RELU	Max	1x1	1x1	0.5
Fully Connected			600				RELU				0.6
Fully Connected							SOFTMAX				0.7

Table 4: Enhanced Model - I

As the model performance is very poor the complete flow is not provided in the appendix.

4.3 Enhanced Model - II

As the second model performance was very poor, we reduced the number of convolution layers in the architecture to two. Now the architecture has two convolution layers and two fully connected layer where the second fully connected layer is the output layer. The architecture of the model is:

The same training setting used for Baseline Model is used for this model too. The only modification here is the number of epochs increased to 20. The model got converged at 5th epoch with 0.96 validation accuracy. The model got 0.95 accuracy in the test data.

Layer	Data	Type	Filters	Kernal	Padding	Strides	Activation	Pooling	Pool Size	Pool Stride	Dropout
Input	Image 32x32	Input Layer									
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.2
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.3
Fully Connected			600				RELU				0.6
Fully Connected							SOFTMAX				0.7

Table 5: Enhanced Model - II

Model	Validation Accuracy	Test Accuracy	Benchmark Accuracy
Baseline Modified - II	0.96	0.95	0.93

Table 6: Accuracy

The model performance exceeds the benchmark of 0.93 in both test and validation data. Still, we made some minor alteration in the architecture to check improvements.

4.4 Enhanced Model - III

This model was created with a minor alteration in the Enhanced Model - II. The base architecture and training settings are same. The only modification is the increase in the number of filters in the fully connected layer. It increased from 600 to 1800. The model got converged at 5th epoch with 0.96 validation accuracy. The model got 0.95 accuracy in the test data.

Model	Validation Accuracy	Test Accuracy	Benchmark Accuracy
Baseline Modified - III	0.96	0.95	0.93

Table 7: Accuracy

The results for Enhanced Model - II and Enhanced Model - III are similar in nature. We finalized this model for the experiment. The final architecture of the model is:

A complete flow diagram of the model is given in the appendix for reference. We have computed multi-class Receiver Operating Characteristics (ROC) for this model. The ROC plot is provided for reference.

5 Testing the Model in Belgium Traffic Sign Images

We tested the model in randomly picked 53 images from the Belgium Traffic Sign Dataset (BTSD). The labels are different in this dataset, so we hand labeled each of these data before evaluation. The accuracy of the model in this data is 0.71, which is

Layer	Data	Type	Filters	Kernal	Padding	Strides	Activation	Pooling	Pool Size	Pool Stride	Dropout
Input	Image 32x32	Input Layer									
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.2
Convolution		Conv2D	32	5x5	valid	1x1	RELU	Max	2x2	2x2	0.3
Fully Connected			1800				RELU				0.6
Fully Connected							SOFTMAX				0.7

Table 8: Enhanced Model - III

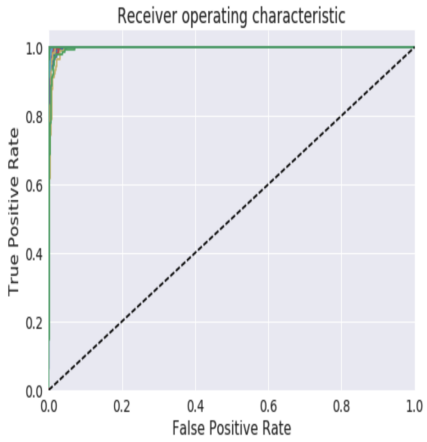


Figure 8: Receiver Operating Characteristics (ROC)

a fair result. The image light conditions and clarity are higher than the GTSBD, and it contains enough perspective variant samples too. Some of the images discussed here along with the top five softmax predictions.

Data	Accuracy	Benchmark	Remarks
Test	0.95	0.93	
External Data	0.71		Belgium Traffic Sign Data

Table 9: Test Data and External Data Accuracy

The first sign we analyzed is two Keep Left signs. The first sign is located in a shady region, and a dark shade is visible in the image. The second image is in a brighter day light, and there is a slight shadow on the top of the image. Also, the second image perspective is slightly different from the first one. Both of the images were accurately recognized as Keep Left. The second closest prediction, in fact too close prediction is Speed Limit 30 km/h and Roundabout Mandatory. The key reason for the prediction Roundabout Mandatory is the color pattern similarity in both of the sign, both shares the same color pattern. The sins and its softmax probability are shown below for reference.

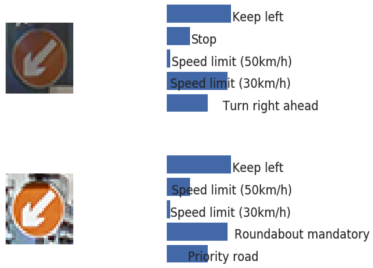


Figure 9: Keep Left Sign Softmax Predictions

The second sign analyzed is the Roundabout Mandatory sign. Two images of the sign were given to the model for testing. The first image was very clean and in bright light, but misclassified as priority road. The similarity between the priority road and roundabout sign is the color pattern. The shape and with in image patterns are different. Still, the image is misclassified. The second sample is taken in the shade and the background is more clear. Surprisingly this image is correctly classified as Roundabout. This particular example shows the possible minor tweaking in the preprocessing may require capturing more image features.

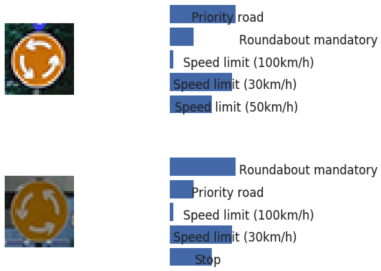


Figure 10: Roundabout Sign Softmax Predictions

The third sign analyzed is the No Entry sign. We took two samples of the sign. In the first image sign orientation is skewed to the left and in the second one, the sign orientation is skewed to the right. Both of the images are taken in almost same bright light conditions. Both were correctly recognized as No Entry sign. The augmentation techniques applied in the training image helped the model to achieve this.

6 Visualizing the Intermediate Results of ConvNet

The intermediate results of a Convolution Network can be captured and visualized. The intermediate layer of a traffic sign image is shown below. The outputs are

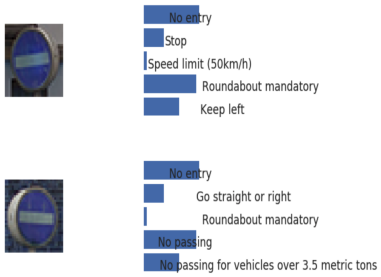


Figure 11: No Entry Sign Softmax Predictions

taken from each stage of first convolution layer in the architecture till applying the dropout.

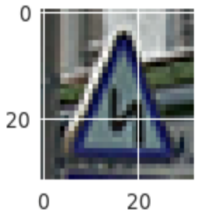


Figure 12: Sign Used for Visualizing the Intermediate Results

7 Discussions

The third model architecture shows significant performance in terms of accuracy and ROC metrics. The performance in a minimal set of external data also shows that the model can perform better on unseen data. The optimal hyperparameters for the model were discovered by adopting an existing architecture and altering it. The only optimizer function tried in this experiment is ADAM. The early stopping strategy with validation loss monitoring helped in reaching the optimal number of epochs for training.

The model can be further enhanced by adopting new augmentation techniques for balancing the image data, trying different preprocessing techniques and different optimizers. A random grid search based training can be performed to find the best optimizer function for this model. Recently many new architectures are proposed for traffic sign classification. A comparison on the performance of such architecture can be done in the next step.

Developing a model for BTSDS with the same model and cross evaluating the model is proposed as potential next step for this exercise.

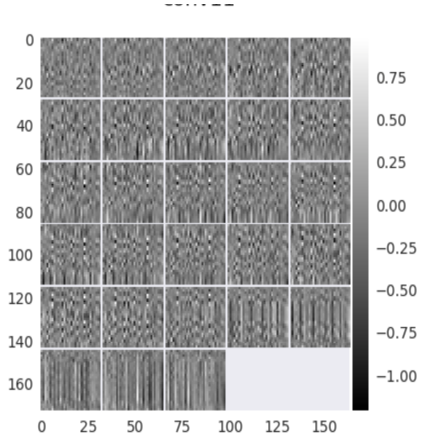


Figure 13: Conv Layer Step - I

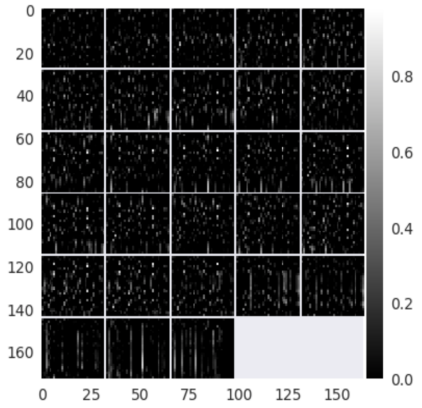


Figure 14: Conv Layer Step - II Activation

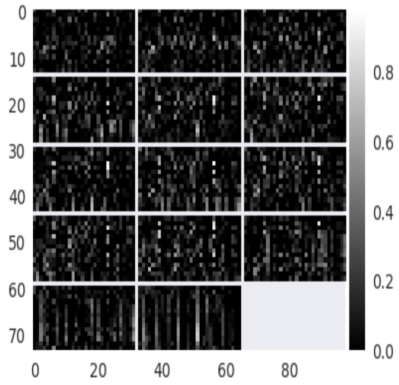


Figure 15: Conv Layer Step - III Max Pool

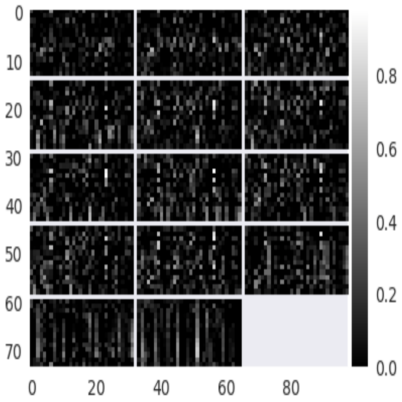


Figure 16: Conv Layer Step - III Dropout