Traffic Sign Recognition - Project 3

Building a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Rubric Points

Data Set Summary & Exploration

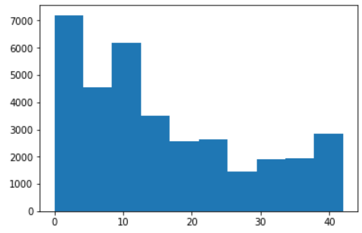
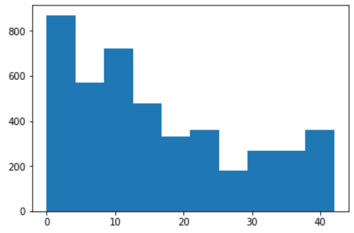
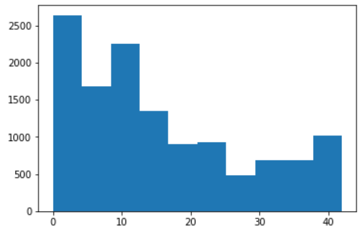
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used the numpy library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32, 32, 3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

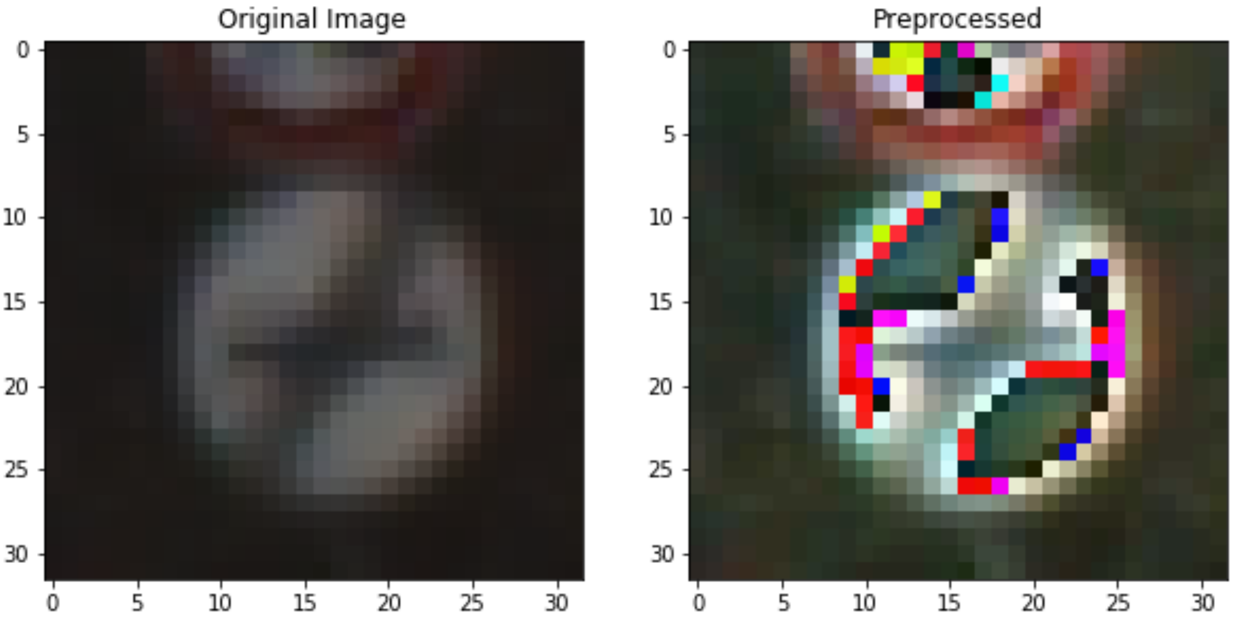
Here is an exploratory visualization of the train, test and validation datasets. It shows the frequency of each class in the respective datasets. It comes as a good sanity check for the fact that the data distribution is approximately the same in all the three sets. Thus, we can consider the validation set as a surrogate for the test set.

Train Histogram	Validation Histogram	Test Histogram
		

Design and Test a Model Architecture

1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques?

I decided to stick with a simple preprocessing step of subtracting each image with the per-channel mean of the training set. The reason to do this is so that the non-important regions (that don't help distinguish) are mostly zero. As an example, refer to the image below:



Since many of the images regularly contained white-filled circle, that part of the image is very close to zero. I also normalize the data by dividing each mean-subtracted image by the standard deviation of the training set. This is done so that the features are at a similar scale and multiplying by weights and adding biases doesn't cause the gradient to explode. Having features at similar scale also aids to the notion of weight sharing - there cannot be weight sharing if the features have a radically different scale.

For the validation set and any new test images, the same mean and standard deviation values are used for preprocessing.

2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model, named `MiniVGG` consists of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 3x3	1x1 stride, same padding, outputs 32x32x64
RELU	
Convolution 3x3	1x1 stride, same padding, outputs 32x32x64
RELU	
Max pooling	2x2 stride, outputs 16x16x64
Convolution 3x3	1x1 stride, same padding, outputs 16x16x128
RELU	
Convolution 3x3	1x1 stride, same padding, outputs 16x16x128
RELU	
Max pooling	2x2 stride, outputs 8x8x128
Fully connected	1024 nodes
RELU	
Fully connected	1024 nodes
RELU	
Fully connected	43 (number of classes) nodes
Softmax	

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

To train the model, I used the Adam Optimizer to minimize the cross-entropy loss. The model achieved a validation accuracy of 93.9% in the 22nd epoch. A batch size of 128 was used along with a learning rate of 0.001.

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated.

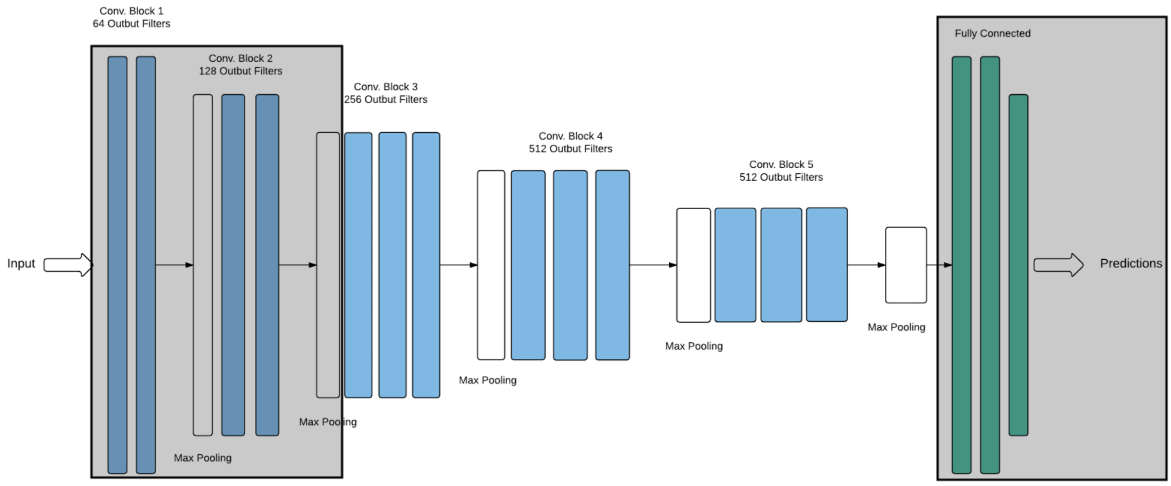
My final model results were:

- training set accuracy of 99.9%
- validation set accuracy of 93.9%

- test set accuracy of 93.3%

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen



  My approach consists of seeking inspiration from a well-known architecture, VGGNet and then iteratively modifying it to suit this dataset. The initial architecture was similar to the one explained above, but it had 4096 nodes in all the fully-connected layers instead of 1024. This value was chosen as it is the number of hidden nodes used in VGGNet as well. Thus, this architecture composed of using the first two layers of VGGNet and the final 3 convolutional layers (as highlighted in the image above). VGGNet has been shown to work well with diverse problem sets and that was the reason I used this as my starting point.

- What were some problems with the initial architecture?

  A major problem with this was that the model was overfitting as the accuracy on the training set kept increasing but that on the validation set kept decreasing. The reason for this is that using 4096 nodes in the hidden layer increases the number of parameters a lot which leads to overfitting.

- How was the architecture adjusted and why was it adjusted?

  In order to reduce the number of parameters, the number of nodes in all the fully connected layers were reduced from 4096 to 1024. This led to a massive reduction in the number of parameters and thus, prevented overfitting.

- Which parameters were tuned? How were they adjusted and why?

  Apart from the number of epochs, the other choices for the optimizer, learning rate and batch size were kept as the same from the LeNet baseline code. Using a learning rate more than 0.001 leads to unstable learning and it is best to keep the learning rate as high as possible such that the training is stable. Also, Adam optimizer uses adaptive gradients, which not only means that the learning rate is adjusted automatically as training progresses, but also that it is adjusted for each parameter separately, which is great in the case of sparsely occuring parameters. The number of epochs were increased from 10 to 50 but the model achieved the desired validation accuracy in 24 epochs itself.

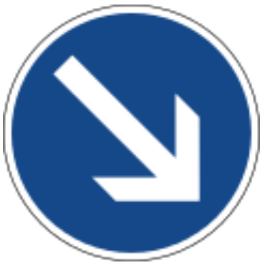
- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

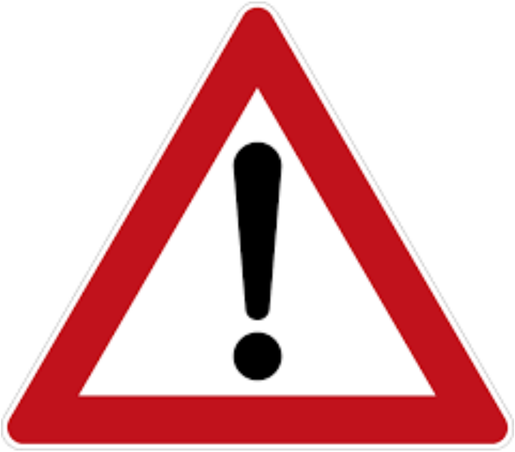
> The most important design choice was to use only 3x3 filters, which shows the simplicity of the design. The other important choice was to successively double the number of feature maps after each block of convolutional operations, instead of after each convolutional operation. This lets us work with a 4-layer model with a much smaller number of parameters. Finally, as mentioned before the choice of the number of the hidden nodes was important for training. It was chosen based on empirical evidence.

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are five German traffic signs that I found on the web:





120

70

The first image might be difficult to classify because it has a transparent background and upon being read, the background is black. The dataset doesn't contain any images with black background. Similarly, the second image consists of no background and the dataset doesn't contain any images with an empty background. The third and fifth both retain the point mentioned above along with an additional difficulty of shadow around the sign. Finally, the fourth image is the most challenging as although it has only one sign, the position of the sign is very different than the model is used to. Also, there are many other things making up the image like roads, houses, etc.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set

Here are the results of the prediction:

Image	Prediction
Keep right	Keep right
General Caution	General Caution
Speed limit (120km/h)	Speed limit (120km/h)
Speed limit (70km/h)	Priority Road
No vehicles	No vehicles

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. Since we are only looking at a sample of 5 images, it is not fair to compare it to the test accuracy of 93.3%. Also, the only image where the model failed was the one which was of a total different distribution that was the model has been seeing. The model is used to seeing the sign at the center and covering most part of the image. When it is reduced to 32x32, the sign barely covers any significant area and hence, the model fails to classify correctly. The interesting part is that even its wrong prediction is not very off - the road comprises of the majority part of the image and the only sign that has a prominent road-like feature is the `Priority Road` sign.

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability.

The code for making predictions on my final model is located below the cell with the heading **Analyze Performance**.

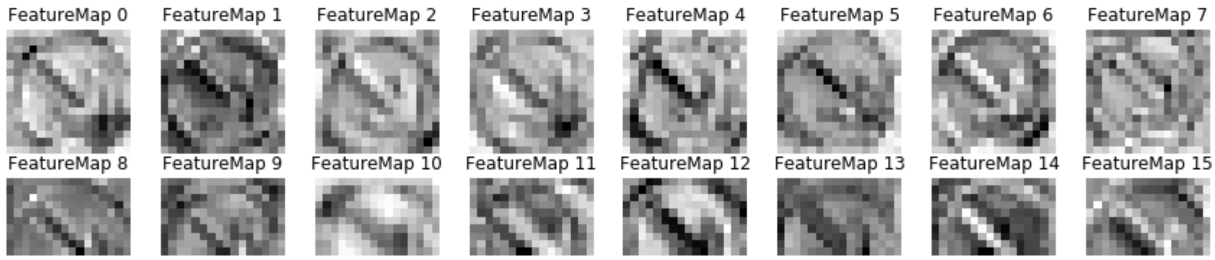
The model is extremely confident about its predictions. For images 1, 2, 3 and 5, it predicts the correct class and assigns it a probability of 0.999 and almost zero to the other classes. It is also very confident of its wrong prediction in the case of image 4:

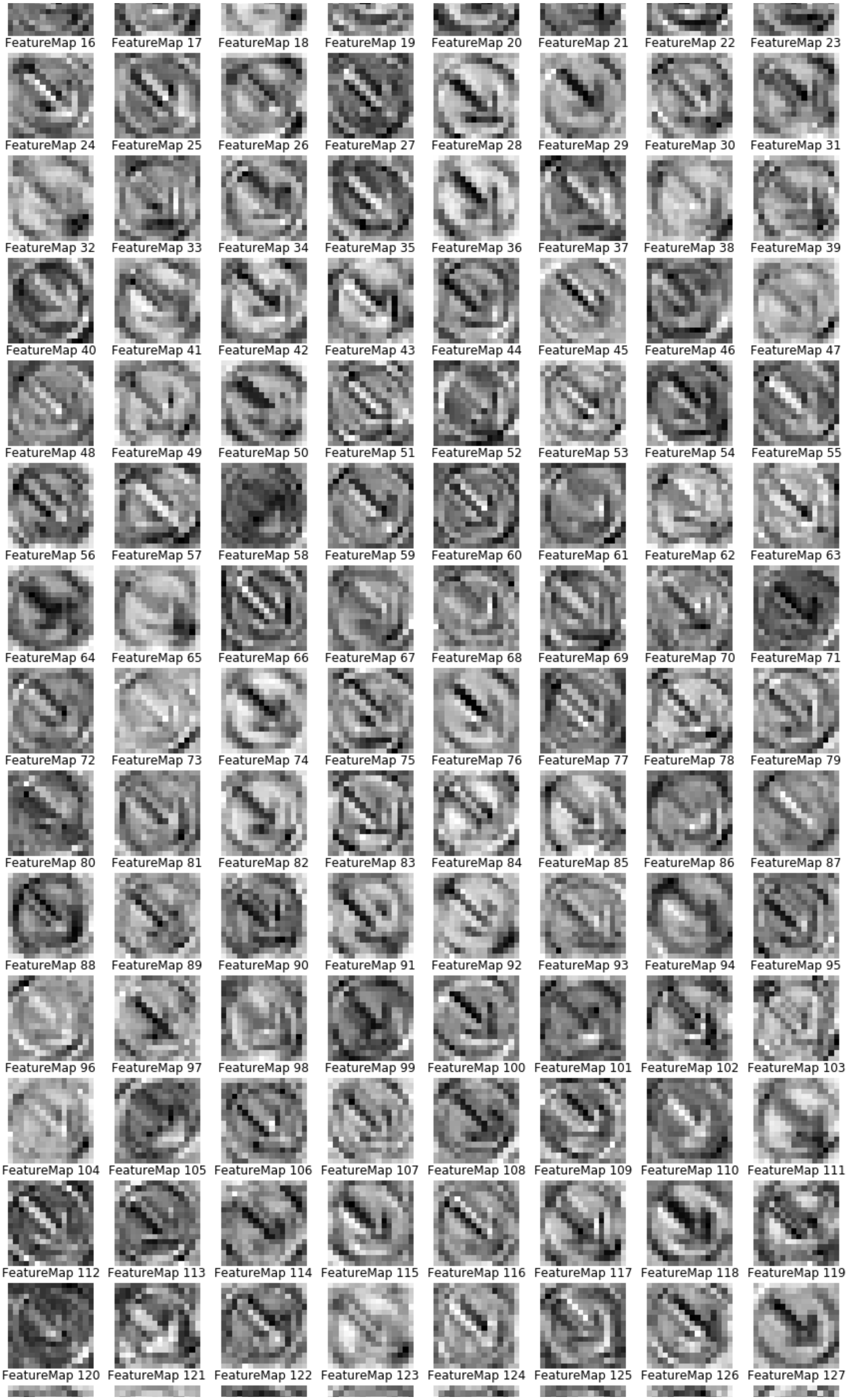
Probability	Prediction
.988	Priority road
.004	Speed limit (80km/h)
.004	Wild animals crossing
.002	Beware of ice/snow
.001	No entry

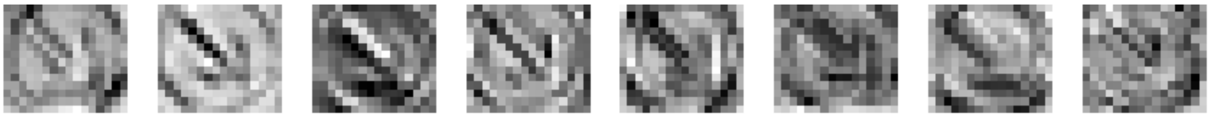
Visualizing the Neural Network

1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

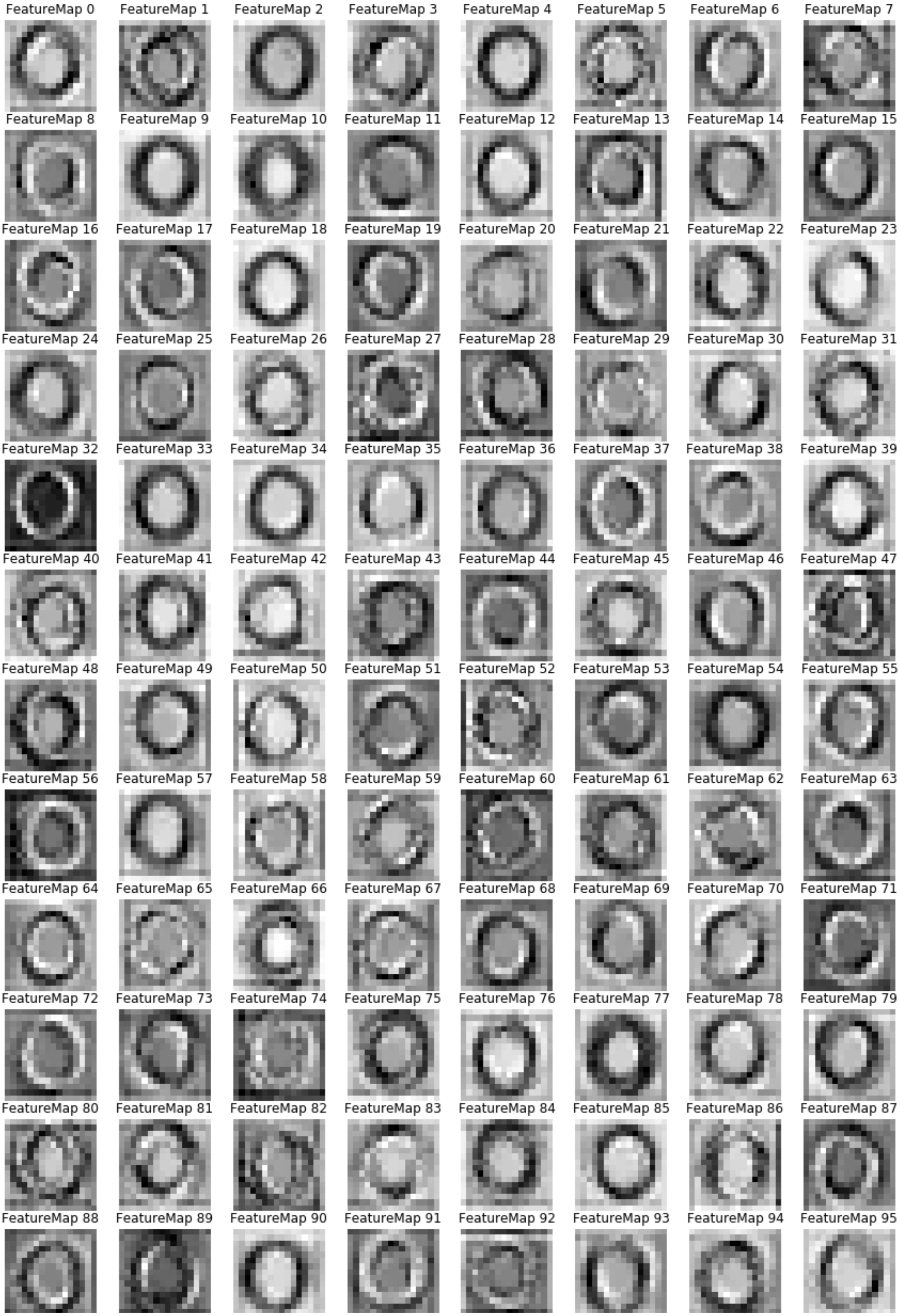
I visualized the outputs of the 3rd convolutional layer (first convolution of the second block) for each of the image. The output consists of 128 feature maps and hence, 128 outputs for each image. It can be seen that certain feature maps are responsive to specific patterns, i.e. they are activated only by that pattern. I specifically discuss images 1 and 5 of the 5 test images chosen by me. The activation for image 1 is shown below:

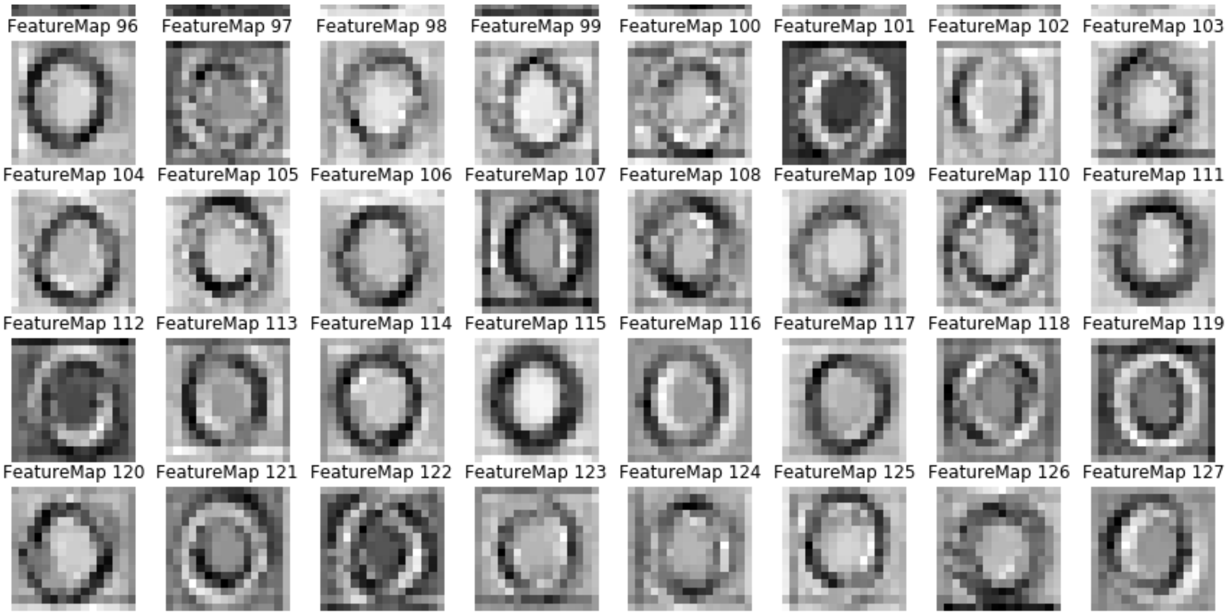



FeatureMap 16
FeatureMap 17
FeatureMap 18
FeatureMap 19
FeatureMap 20
FeatureMap 21
FeatureMap 22
FeatureMap 23
FeatureMap 24
FeatureMap 25
FeatureMap 26
FeatureMap 27
FeatureMap 28
FeatureMap 29
FeatureMap 30
FeatureMap 31
FeatureMap 32
FeatureMap 33
FeatureMap 34
FeatureMap 35
FeatureMap 36
FeatureMap 37
FeatureMap 38
FeatureMap 39
FeatureMap 40
FeatureMap 41
FeatureMap 42
FeatureMap 43
FeatureMap 44
FeatureMap 45
FeatureMap 46
FeatureMap 47
FeatureMap 48
FeatureMap 49
FeatureMap 50
FeatureMap 51
FeatureMap 52
FeatureMap 53
FeatureMap 54
FeatureMap 55
FeatureMap 56
FeatureMap 57
FeatureMap 58
FeatureMap 59
FeatureMap 60
FeatureMap 61
FeatureMap 62
FeatureMap 63
FeatureMap 64
FeatureMap 65
FeatureMap 66
FeatureMap 67
FeatureMap 68
FeatureMap 69
FeatureMap 70
FeatureMap 71
FeatureMap 72
FeatureMap 73
FeatureMap 74
FeatureMap 75
FeatureMap 76
FeatureMap 77
FeatureMap 78
FeatureMap 79
FeatureMap 80
FeatureMap 81
FeatureMap 82
FeatureMap 83
FeatureMap 84
FeatureMap 85
FeatureMap 86
FeatureMap 87
FeatureMap 88
FeatureMap 89
FeatureMap 90
FeatureMap 91
FeatureMap 92
FeatureMap 93
FeatureMap 94
FeatureMap 95
FeatureMap 96
FeatureMap 97
FeatureMap 98
FeatureMap 99
FeatureMap 100
FeatureMap 101
FeatureMap 102
FeatureMap 103
FeatureMap 104
FeatureMap 105
FeatureMap 106
FeatureMap 107
FeatureMap 108
FeatureMap 109
FeatureMap 110
FeatureMap 111
FeatureMap 112
FeatureMap 113
FeatureMap 114
FeatureMap 115
FeatureMap 116
FeatureMap 117
FeatureMap 118
FeatureMap 119
FeatureMap 120
FeatureMap 121
FeatureMap 122
FeatureMap 123
FeatureMap 124
FeatureMap 125
FeatureMap 126
FeatureMap 127



The activations for image 5 is shown below:





It can be clearly seen that this layer is looking for shapes. Since most of the traffic signs are circular, most of the feature maps also get activated by the presence of a circle, clearly visible for the visualization corresponding to image 5. However, for the visualization corresponding to image 5, it can be clearly seen that some feature maps contain the same value throughout signifying that they are not activated, whereas certain feature maps are strongly activated at the presence of the downward right arrow. Thus, the layers are learning the right features.