DATASET SUMMARY

The dataset summary is shown as the output of the first notebook cell.

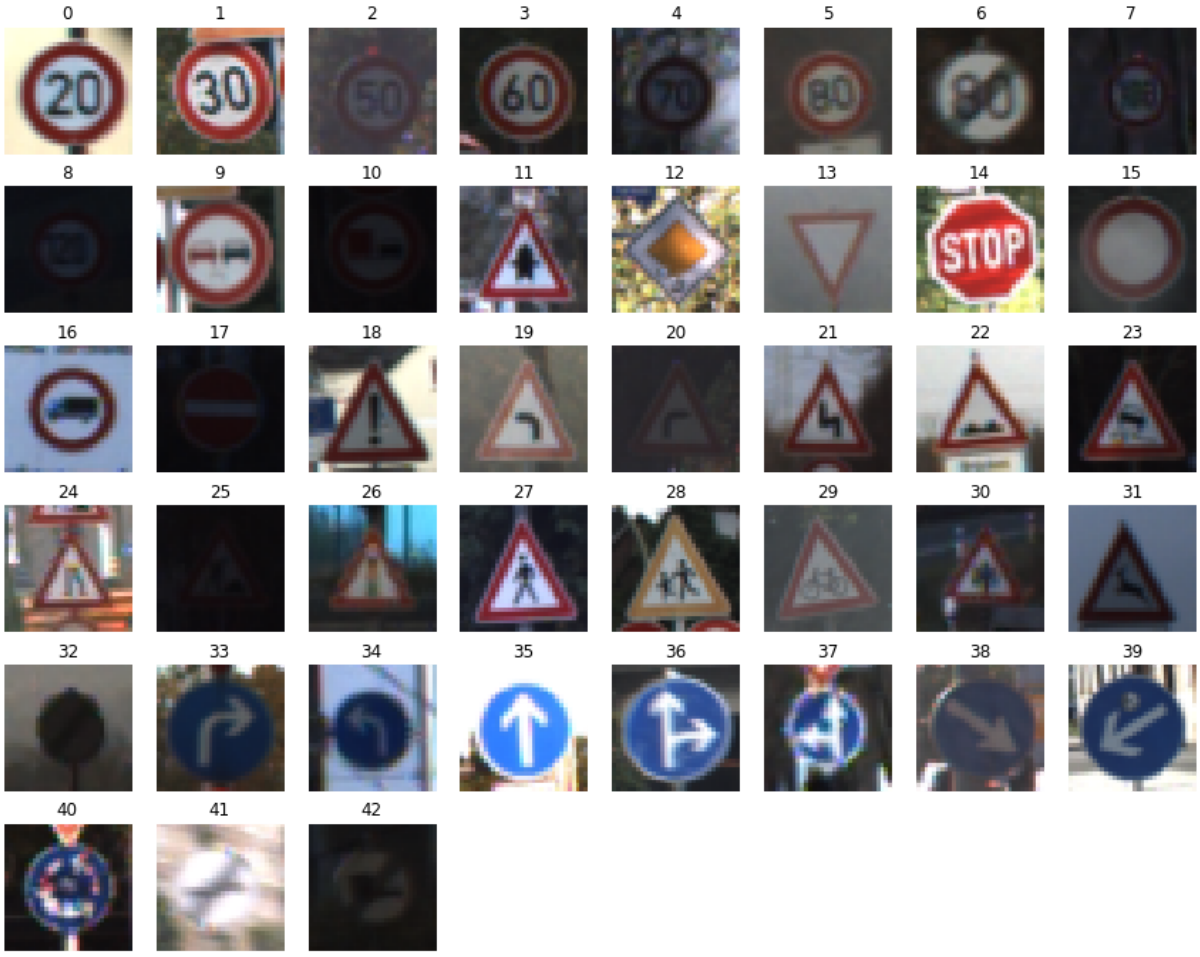
Eg output -

Number of training examples = 34799
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43

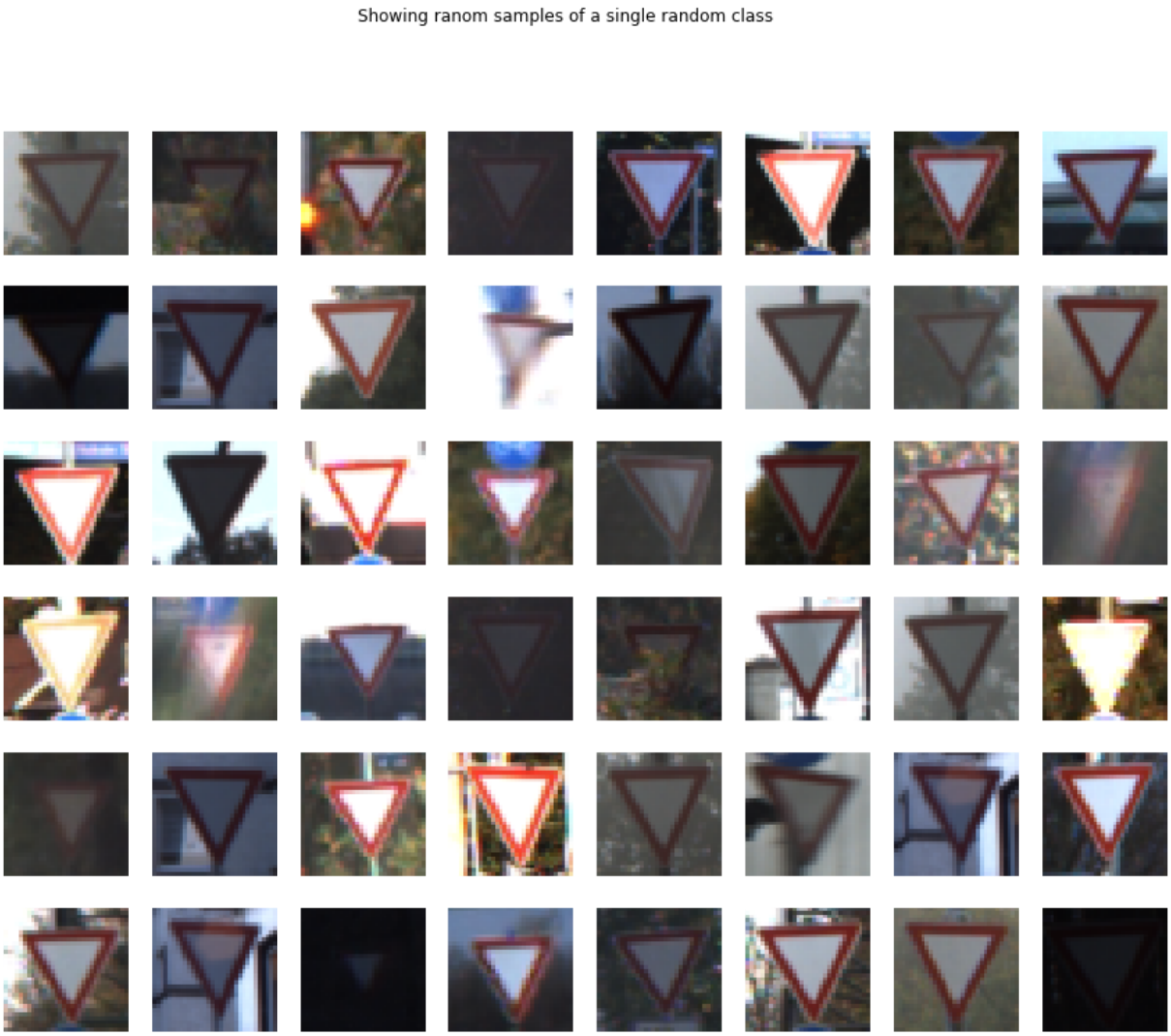
EXPLORATORY VISUALISATION

The next code cell implements code to visualise any random image of each of the classes

Eg output -



Another visualisation provides 48 random samples of a random class -
Eg output -

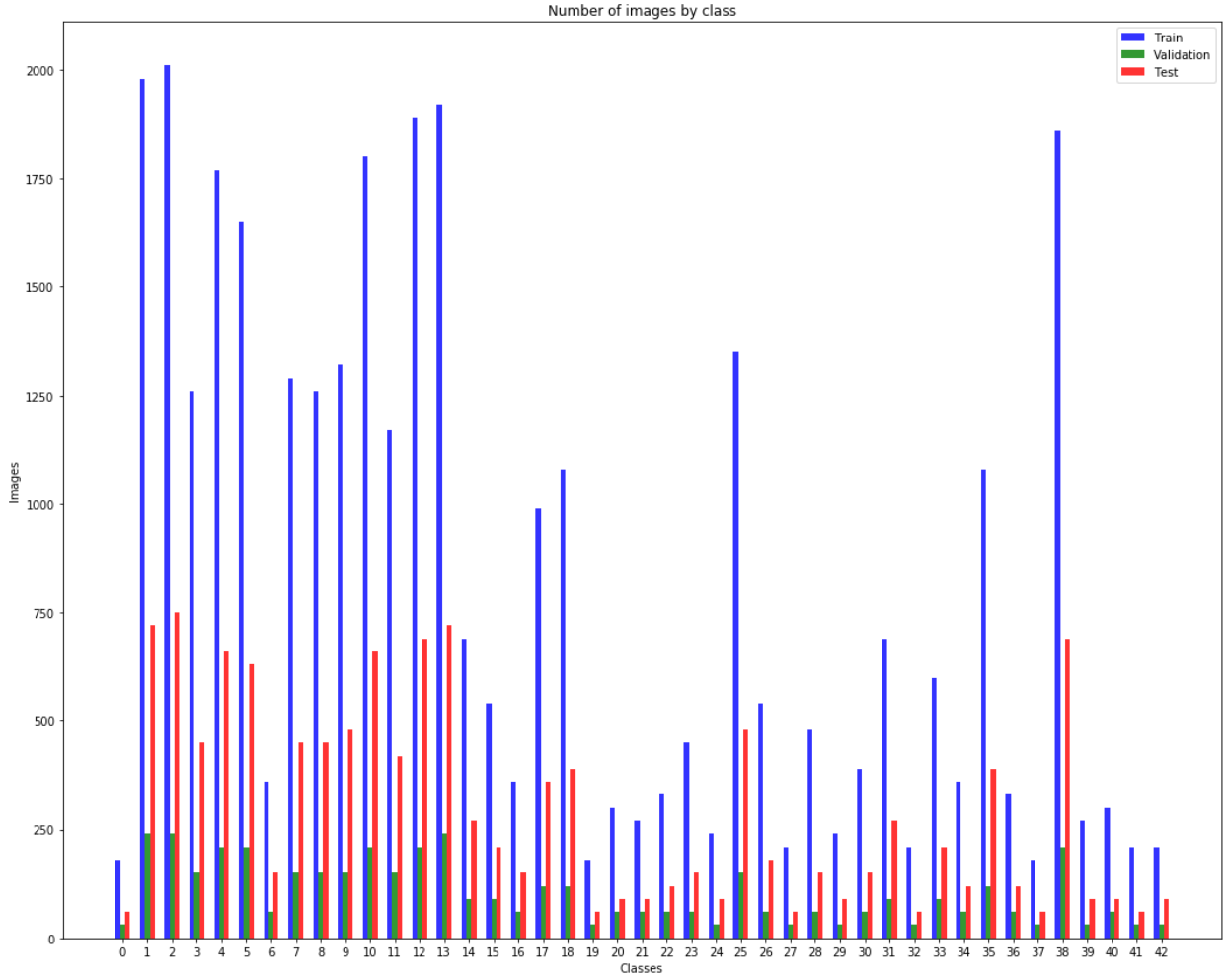


Some image properties -

- We can see a high variation of brightness and contrast in the images
- Some images are even blurred and jittered
- The images have varied backgrounds also
- I could not find any image that contained more than one traffic sign

Another visualisation provides a distribution over the classes, from all the three train, validation and test sets

Eg output -



PREPROCESSING

Images are normalised so that all the features are of the same scale. This helps all of them to have an equal vote in prediction. Also helps gradient steps to converge faster.

Eg output -

Before Processing, Mean and Std Deviation of the channels

[86.69812205 79.49594061 81.83870445] [69.28387882 66.26993615 67.76634548]

After Processing, Mean and Std Deviation of the channels

[1.3454047e-06 2.4697260e-05 -1.1277525e-06] [0.9432926 0.9388118 0.9365145]

MODEL ARCHITECTURE

The Lenet architecture was modified by adding more filters on each convolution layer and adding 1 more convolution layer. Make the model more heavy would mean overfitting which is dealt by introducing dropout.
In a general rule of thumb I have tried to make the model heavy and then introduced dropout.
The architecture looks like -

1. Conv1 (5x5, stride=1, filters=32) (In, Out) = (5x5x3, 28x28x32)
2. Max Pool (2x2, stride=2) (In, Out) = (28x28x32, 14x14x32)
3. Relu
4. Conv2 (3x3, stride=1, filters=64) (In, Out) = (14x14x32, 12x12x64)
5. Max Pool (2x2, stride=2) (In, Out) = (12x12x64, 6x6x64)
6. Relu
7. Conv3 (3x3, stride=1, filters=128) (In, Out) = (6x6x64, 4x4x128)
8. Relu
9. Flatten (In, Out) = (4x4x128, 2048)
10. Fully Connected , (In, Out) = (2048, 512)
11. Relu
12. Dropout
13. Fully Connected , (In, Out) = (512, 256)
14. Relu
15. Dropout
16. Fully Connected (In, Out) = (256, 43)

TRAINING

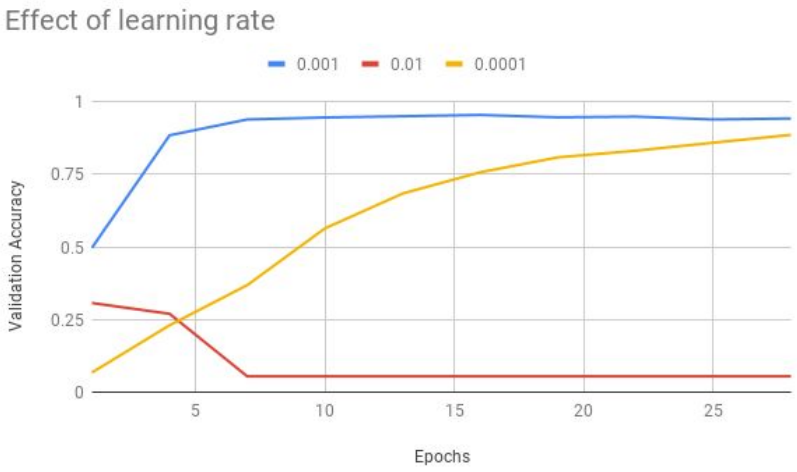
The training is exactly the same as used in the LeNet lab in lectures

Optimizer - Adam

The choice of optimizer is inspired from the cs231n lecture on different optimisation methods.
It has currently become the default first choice of most deep learning practitioners.
Using SGD would require us to adapt the learning rate manually as training proceeds (lower it in later stages). Adam on the other hand is able to adapt the learning rate for each parameter.

Learning rate - 0.001

I experimented with a few learning rates to boil down to the final value.
A value of 0.01 results in exploding gradients and a value of 0.0001 results in very slow training
A small visualisation is shown below -



Epochs - 30

Number of epochs is set as 30, as over multiple runs it was noticed that the validation accuracy(given the learning rate of 0.001) stops increasing after 20.

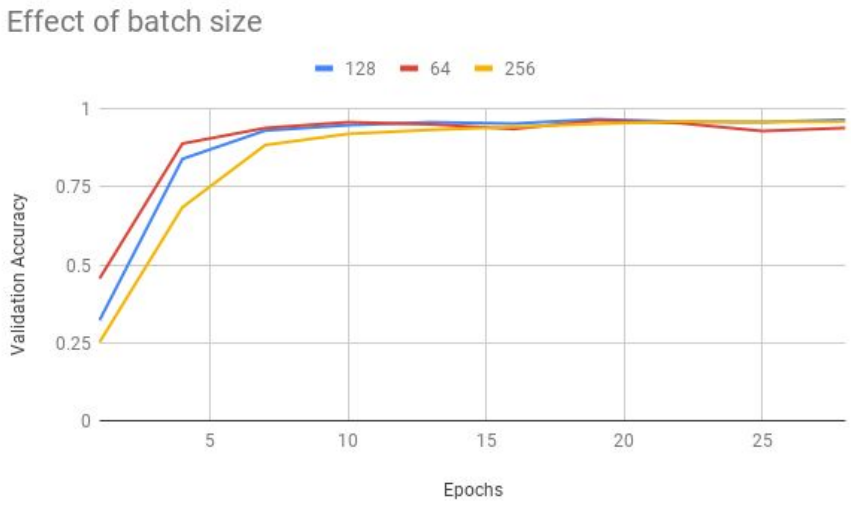
Batch Size - 128

Using a batch size of 128 is inspired from what deep learning practitioners generally set the value to be. The values 64 and 128 was also experimented but yielded no drastic difference in validation accuracy.

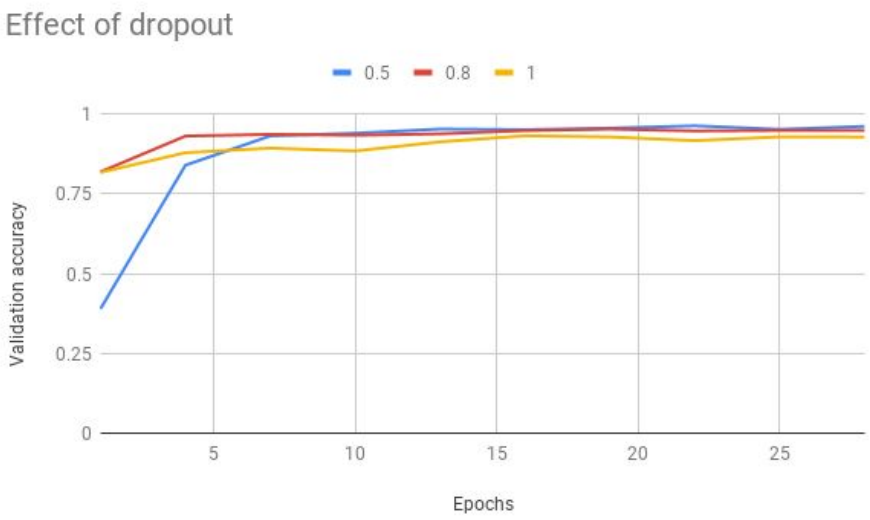
A very low batch size although does increase training time as more gradient steps happen per epoch. It also slows down convergence due to increased stochasticity in the gradient steps. This was noticed with a batch size of 8

A very high batch size cannot be used due to memory constraints although with images of just 32 by 32 we can push higher.

A small visualisation is shown below -



Dropout keep probability - 0.5
Lower dropout values led to faster convergence but resulted in a hit on validation and test accuracy.
The values 0.5, 0.8 and 1.0(no dropout) were experimented with
Visualisation -



SOLUTION APPROACH

The solution approach is similar to the one defined in the model architecture section.
First I started off with the LeNet architecture. Got a test accuracy of around 0.89.
After introducing dropout(0.5) in LeNet, the validation and test accuracy went up to 0.93
To improve upon it further the model is made bigger and deeper.

The model is a bit inspired from ZFNet as that is the first model I used for transfer learning in a different project. Some of the inspirations include

- Setting a higher kernel size in the early layers and then gradually reducing it
- Setting the number of filters to be powers of 2 and gradually increasing it as we go deeper

The idea was to boil down to something bigger than LeNet but not as big as ZFNet due to small images. Dropout is then introduced in the fully connected layers for regularisation.
More number of epochs are thus needed due to high dropout.
I also experimented with slight changes in keep probability and learning rate, as shown above to get the final hyperparameters.

We measure the model's performance by a simple ratio of the predictions it gets right, to the total predictions on the test set. This test set is neither shown during training neither is used to tune hyperparameters, thus does not bleed information into training
We get a validation and test accuracy of around 0.95

ACQUIRING NEW IMAGES

New images are acquired through a simple google image search. Have tried to avoid getting images from web pages which have the same images as in the training dataset.
The images are all of a white background, something I did not observe in the training dataset when I was visualising it.
Some image properties -

- The images are quite bright and have good contrast to be able to clearly differentiate the sign from the background
- All images have a white background
- The images are quite clear and are not jittered
- In cases where the traffic signs are recognised by angles they are pretty close to that of the training data
- One image has only one traffic sign

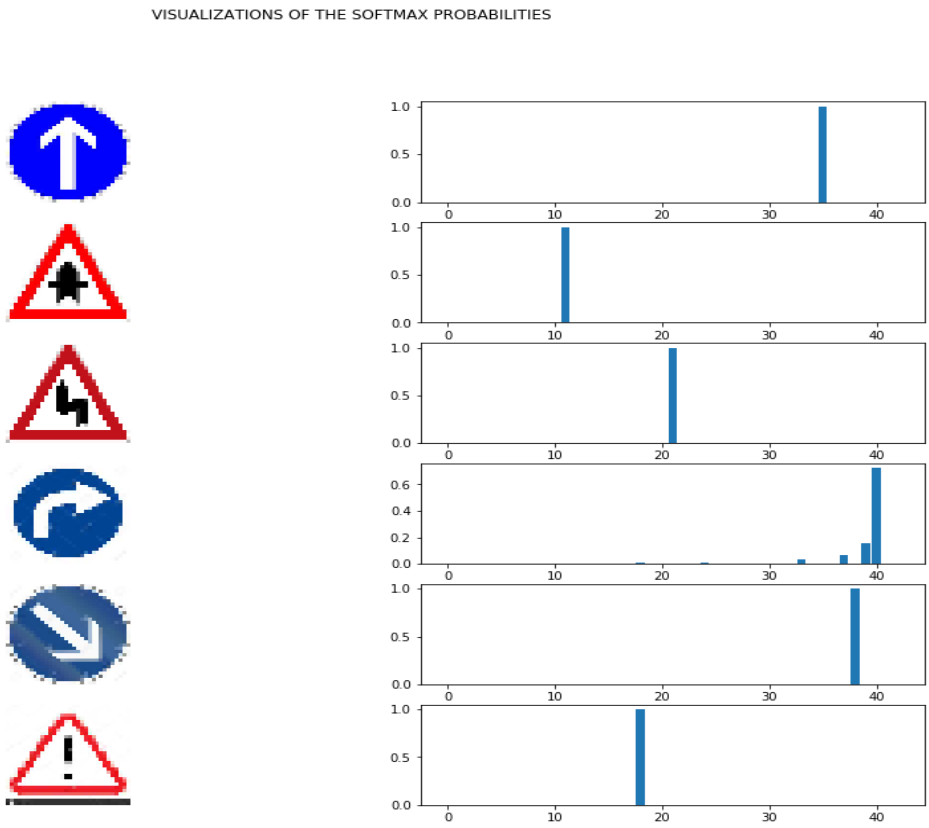
PERFORMANCE ON NEW IMAGES

The model gets 5 out of the 6 images rightly classified resulting in a accuracy 0.833
This is quite low compared to the test set accuracy of 0.95 but benchmarking against only 6 images from a different domain than the test set, is something we cannot rely on. We would need more new images to actually draw any decent conclusions

MODEL CERTAINTY - SOFTMAX PROBABILITIES

The softmax probabilities of all the predictions are then plotted as a bar chart and visualised. An important point to note is that the probabilities are more diffused on the image it gets wrong compared to other, signalling its not very sure about its prediction

Eg output -

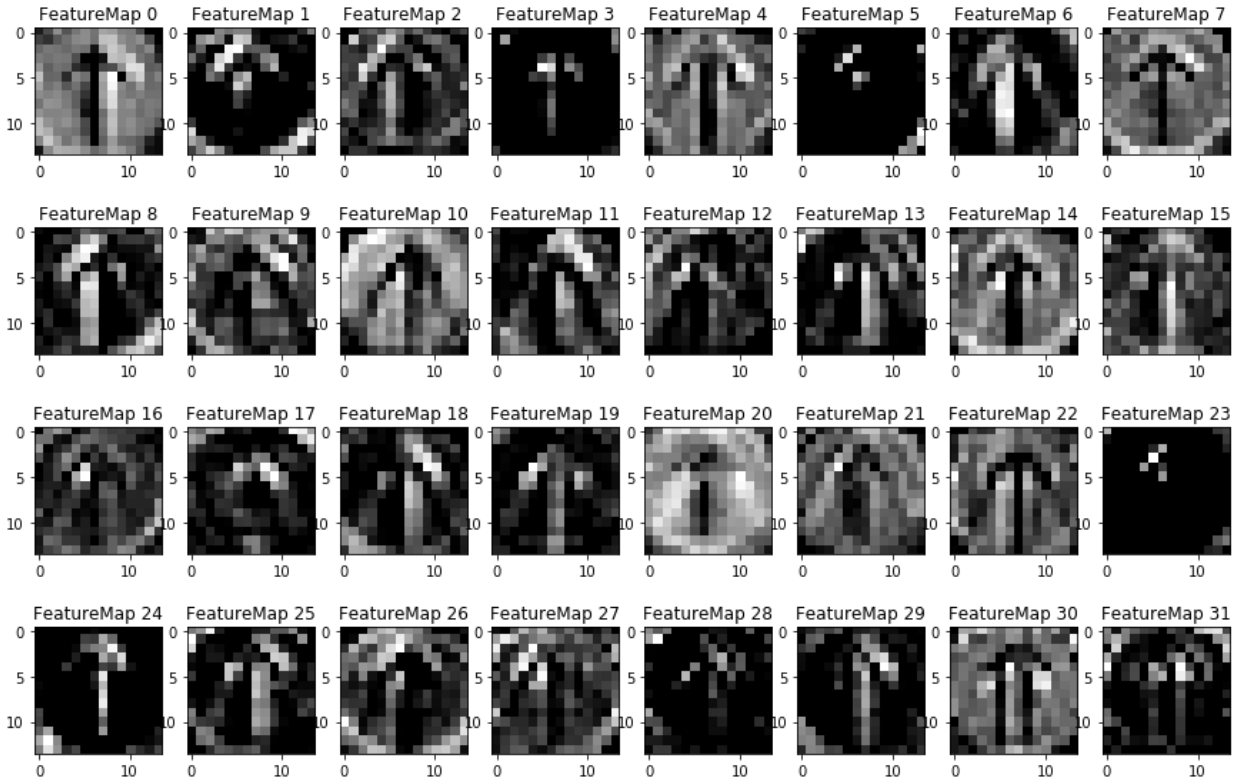


VISUALISING FEATURE MAPS

The activations resulting from the first convolution layer is then visualised.

Image -

Activations -



We can clearly see some sort of edge detection going on with some filters more sensitive to curved edged and some straight.

Some are showing sensitivity to different colors as well