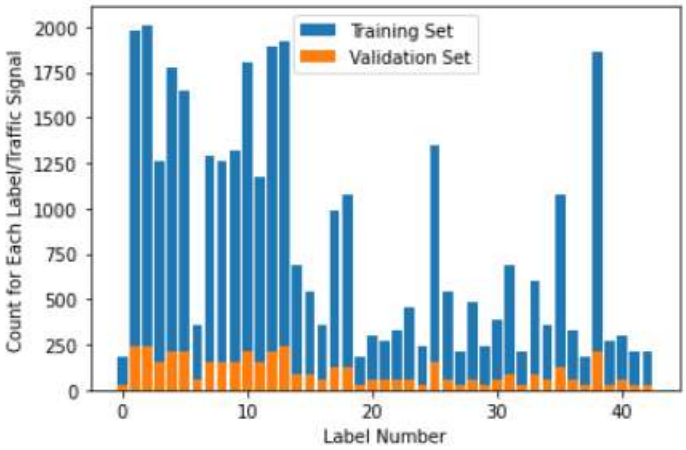
Traffic Sign Classifier

Dataset Exploration

- Dataset Summary
  Number of training examples: 34799
  Number of validation examples: 4410
  Number of testing examples: 12630
  Input image shape: (32, 32, 3)
  Number of output classes: 43

- Exploration Visualization
  The below histogram shows the number of examples for each output label both from the testing set and the validation set to visualize the relative difference in each set:



The below picture shows a random set of 10 images along with the assigned label number from the training set to visualize the differences in images:

```
[25972, 10781, 29727, 2395, 13477, 11815, 7250, 29866, 34296, 24137]
```

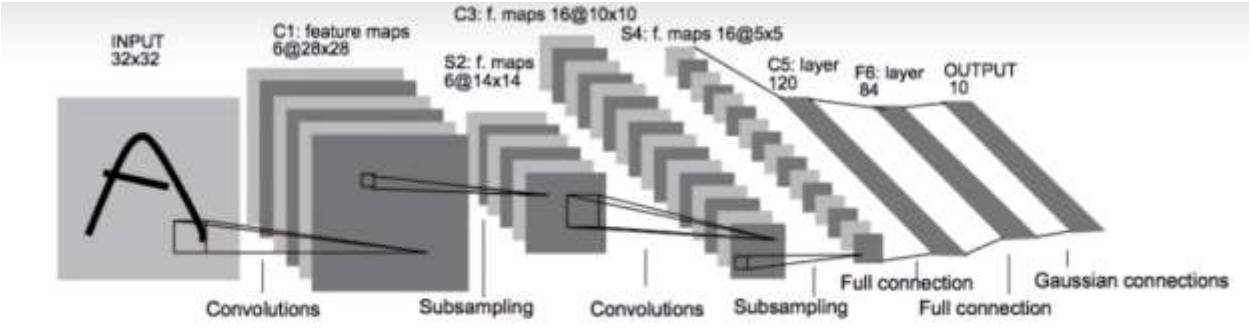
Design and Test Model Architecture

- Preprocessing
  Preprocessing is a very important step in developing a convolutional neural network as we want the inputs to be on the same scale
  Step 1: Mean for all the channels was calculated to ensure that light intensity did not play major role in the learning process and in general grayscale images provide better results

  Step 2: The images were normalized to ensure that the data has a mean of zero and equal variance and hence ensure the images are on a similar scale

  Step 3: The images were shuffled to ensure that the model did not learn the order of images

- Model Architecture
  The starting point for the model was the LeNet5 and the model immediately gave an accuracy of 89% on the validation test. To increase the accuracy a dropout layer was added after the second fully connected layer and initially a keep probability was kept at 0.5. this layer helped in reducing the chances of overfitting. The picture below is from the LeNet5 developed by Yan LeCun. With respect to this, the model for this training has another layer after the second fully connected layer which uses dropout to reduce overfitting. Also, the number of output labels in our model is 43 instead of 10.



Source: Yan LeCun

The table below give the input and output sizes for all the layers, along with the activation function and type of padding used

S.No.	Layer	Input Size	Output Size
1	5x5 Convolutional Layer Stride: [1, 1] Padding: Valid	Image 32x32x1	28x28x6
2	Activation Function: RELU	28x28x6	28x28x6
3	Max Pooling Kernel size: [2, 2] Stride = [2, 2]	28x28x6	14x14x6
4	5x5 Convolutional Layer Stride: [1, 1] Padding: Valid	14x14x6	10x10x16
5	Activation Function: RELU	10x10x16	10x10x16
6	Max Pooling Kernel size: [2, 2] Stride: [2, 2]	10x10x16	5x5x16
7	Flatten Layer	5x5x16	400
8	Fully Connected	400	120
9	Activation Function: RELU	120	120
10	Fully Connected	120	84
11	Activation Function: RELU	84	84
12	Dropout Layer	84	84
13	Fully Connected Output Layer	84	43

- Model Training

  Adams Optimizer was used it is one of the most computationally efficient optimizers and has little memory requirements. It is a stochastic gradient descent optimizer
  Batch size was kept at 128. Epochs were increased to 100 initially to see if the model was overfitting and slowly reduced to 50 as the model was converging and achieving an accuracy of more than 93%
  Other hyperparameters are mentioned in the table below:

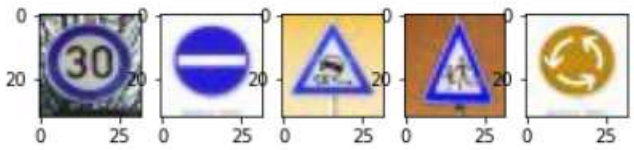
Hyperparameter	Variable Name	Value
Learning Rate	rate	0.001
Epochs	epochs	50
Batch size	batch	128
Keep probability	keep_prob	0.7

Mean of weights	mu	0
Sigma of weights	sigma	0.1

- Solution Approach
  The above hyperparameters were drastically increased and decreased to understand how they impacted the result while keeping the learning rate low and keeping the epochs close to 100. Once the architecture and the other hyperparameters gave satisfactory results, the learning rate was slowly increased, and the epochs were reduced
  The model gave validation accuracy of 94.4% and test set accuracy of 91.9%

Test a Model on New Images

- Acquiring New Images
  The following images were downloaded from the internet and converted to 32x32 using MS Paint. As these images were converted to smaller pixels, some of the details maybe lost and hence it could be difficult for the model to classify them correctly



- Performance on New Images
  The model achieved an accuracy of 0.8 on the downloaded images as compared to an accuracy of 0.944 on the test set provided. But this accuracy could vary if the images were not clear enough

- Model Certainties – Softmax Probabilities

```
INFO:tensorflow:Restoring parameters from ./lenet
TopKV2(values=array([[9.99999881e-01, 1.02738014e-07, 3.99582700e-10, 5.2660955
5e-11,
        2.55827443e-11],
       [9.99999881e-01, 1.16982790e-07, 2.37826148e-10, 4.25931568e-14,
        2.40105154e-17],
       [9.40245688e-01, 2.95954701e-02, 2.87118424e-02, 9.84704238e-04,
        4.03189857e-04],
       [1.00000000e+00, 9.56782875e-10, 1.27988592e-11, 8.57621092e-12,
        3.15923405e-15],
       [9.99999642e-01, 3.70805225e-07, 2.22822607e-12, 8.57511897e-14,
        2.80224909e-14]], dtype=float32), indices=array([[ 1, 38,  2,  3,  5],
       [17,  9, 34, 12, 40],
       [23, 28, 19, 29, 20],
       [28, 24, 20, 29, 30],
       [40,  9, 12, 41, 17]]))
```