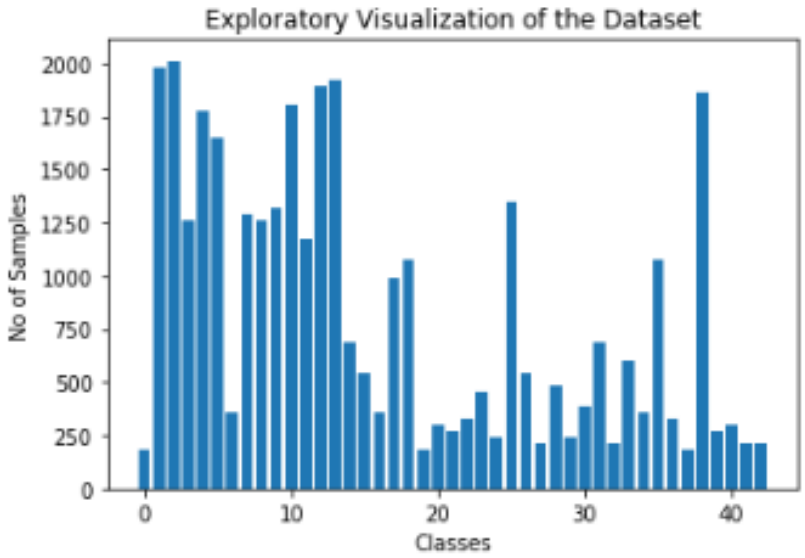
**Traffic Sign Recognition**

1. **Dataset Summary and Exploration**
   a. **Basic Summary**
   Using numpy to provide a basic summary and statistics for the data set:

   Number of training examples = 34799
   Number of validation examples = 4410
   Number of testing examples = 12630
   Shape of traffic sign image = (32, 32, 3)
   Number of unique classes/labels = 43

   b. **Exploratory Visualization**
   Here is a bar chart showing the number of samples in each class in the data set:

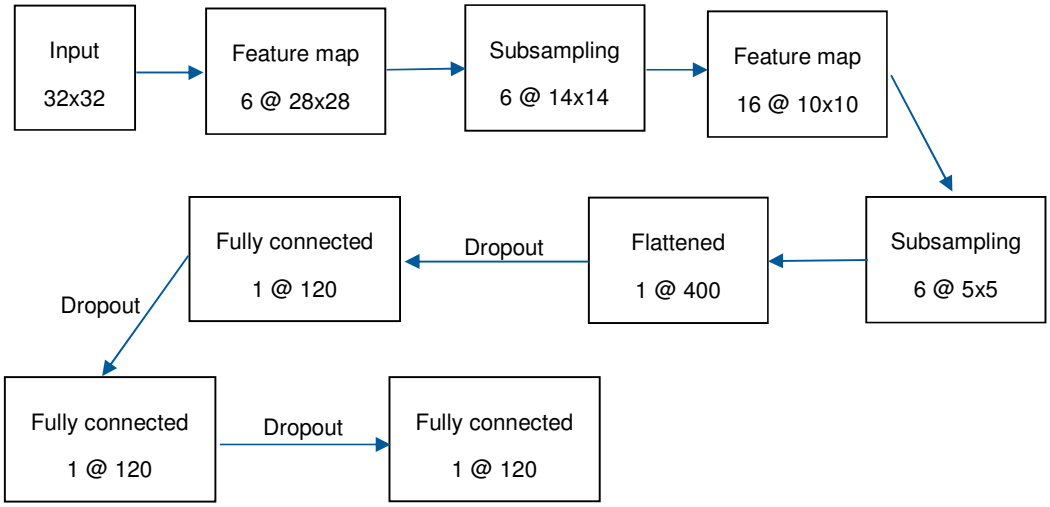



2. **Design and Test a Model Architecture**
   a. **Preprocessing techniques**
   I chose to preprocess the data in two steps: grayscale and normalization. As we learned in previous lessons, grayscaling reduces the amount of features, which reduces execution time. For normalization, I used the suggested formula of (pixel - 128) / 128 to normalize the data from (0,255) to (-1, 1).

   b. **Final model architecture**
   My final model is based on the LeNet architecture. This model has dropout layers in between each fully connected layer. This helps to avoid overfitting.

Here is a table showing all the layers in this model:

| LAYER | DESCRIPTION | INPUT | OUTPUT |
|---|---|---|---|
| Input | Inputs a grayscaled image | 32x32x1 | |
| Convolutional 5x5 | 1x1 stride, valid padding | 32x32x1 | 28x28x6 |
| RELU | | | |
| Pooling | 2x2 stride, valid padding | 28x28x6 | 14x14x6 |
| Convolutional 5x5 | 1x1 stride, valid padding | 14x14x6 | 10x10x16 |
| RELU | | | |
| Pooling | 2x2 stride, valid padding | 10x10x16 | 5x5x16 |
| Flatten | | 5x5x16 | 400 |
| Dropout | | | |
| Fully connected | | 400 | 120 |
| RELU | | | |
| Dropout | | | |
| Fully connected | | 120 | 84 |
| RELU | | | |
| Dropout | | | |
| Fully connected | | 84 | 43 |



**c. Training the model**

The model was trained using Adam Optimizer. Following hyperparameters were used:
Epochs: 30
Batch size: 128
Dropout: 0.5 (keep probability of the dropout layer)
Learning rate: 0.001
Arguments used for truncated normal distribution mu = 0, sigma = 0.1

**d. Validation set accuracy**

Training set accuracy: 98.2%
Validation set accuracy: 93.8%
Test set accuracy: 92.4%

**e. Solution approach**

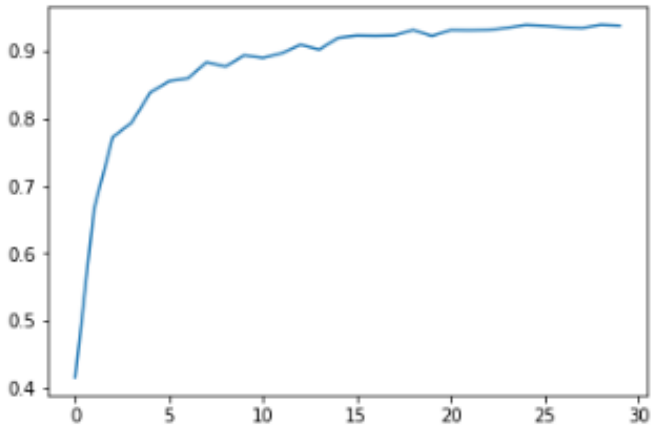
I started with the LeNet architecture model from the lesson, and the first thing I did is to modify it to output to match the 43 unique classes in the training data set.
The hyperparameters I used were: Epochs = 15, batch size = 128, learning rate = 0.0009, mu = 0, sigma = 0.1)
However, the accuracy was lower than 85%, as expected. I tried this on my test images, and for “Go straight or turn right” it actually detected it as “Go straight or turn left” !

Preprocessing to add grayscaling and modifying the input to 32x32x1 helped improve accuracy but I still could not get better than 91%.

Increasing the number of epochs to 25 helped improve accuracy of the training set to 95%, however validation set accuracy was still around 91%, which is lower than what was expected, and indicates overfitting.

To reduce the overfitting, I added dropout layers before each fully connected layer. This helped reduce overfitting, however I observed some inconsistencies during epochs 22 to 25, which sometimes resulted in the validation set accuracy going above 93% but finally ending at 92.8%, which is not good enough.

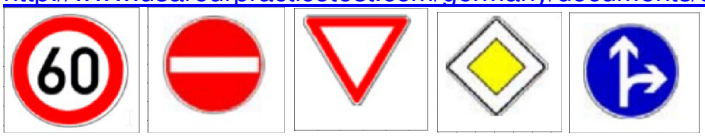
Then I changed the learning rate to 0.001 and number of epochs to 30, to get consistent results for accuracy.



## 3. Test a Model on New Images

### a. Signs chosen

I downloaded German traffic sign chart from here and took screenshots of images.
http://www.usareurpracticetest.com/germany/documents/sign_chart.pdf



As most speed signs are similar, i.e. number inside a red circle, there may be difficulty predicting the correct number.
The last sign, i.e. “Go straight or turn right” is similar to “Mandatory right turn” or also similar to “Mandatory go straight”.
“Priority road” and “End of priority road” signs are very similar as well.

### b. Discuss the model’s predictions

I first resized all these images to 32x32 before running them through the model.
The model correctly predicted 4 out of 5 signs. The one it failed at was the “Speed limit 60km/h” sign, which it identified as “Speed limit 50km/h”.
This is due to loss of resolution due to the quality of the image, resizing the image, and similarity to other speed limit signs.
Accuracy could be improved by augmenting the training data with more images in various orientations like rotated, zoomed, images with graffiti, blurred, partially obstructed by tree branches, etc.
Another way would be to preprocess better and not lose so much resolution.

### c. Softmax probabilities for each prediction

The bar graph below shows the top 5 softmax probabilities of the predictions. As seen in the bar graph, softmax predictions for the correct top prediction is greater than 95%.