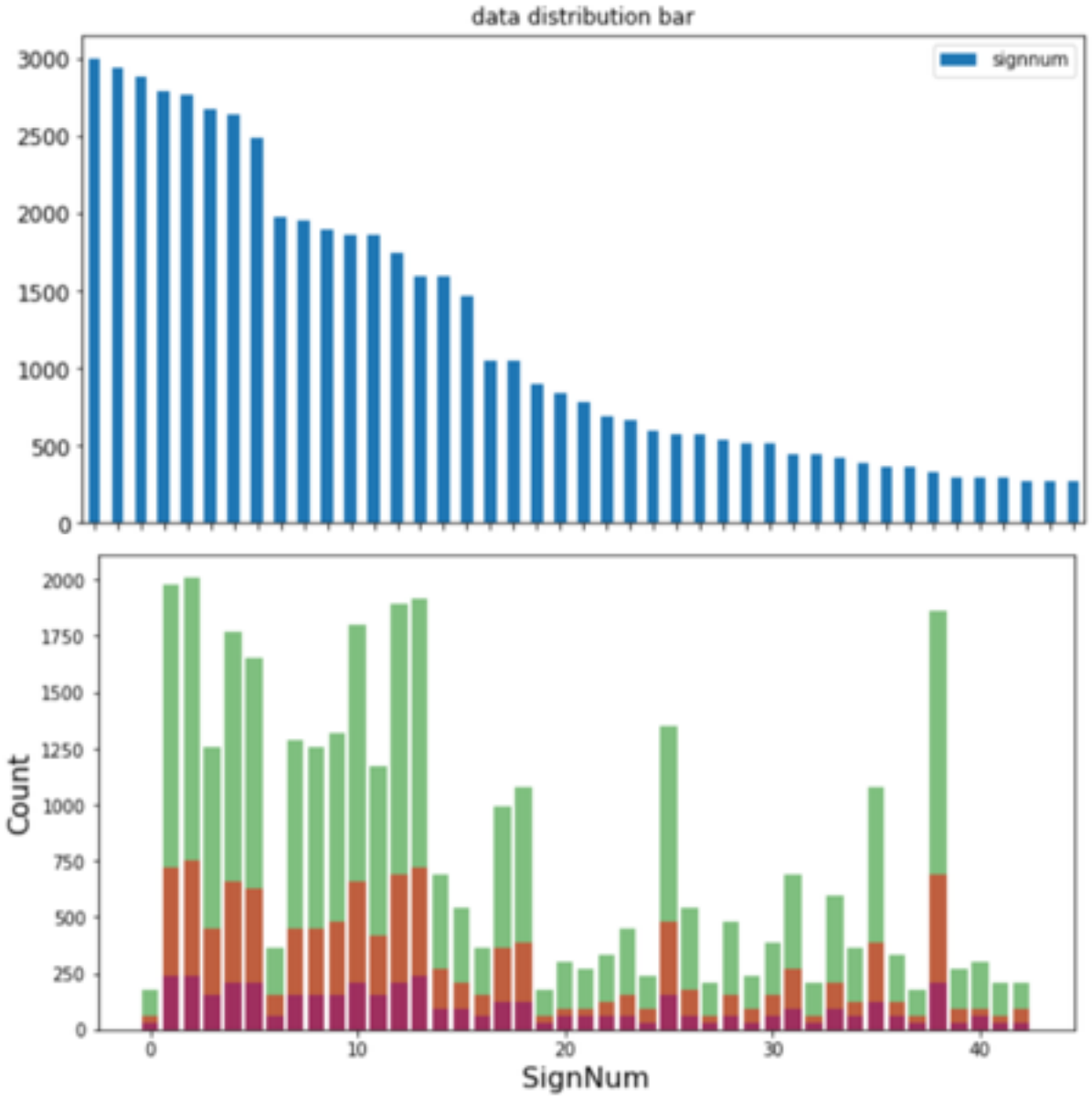
Traffic Sign Classifier

1. Dataset Summary

```
Training Set:   34799 samples
Validation Set: 4410 samples
Test Set:       12630 samples
Dataset shape is (32,32,3)
```

2. Exploratory Visualization

Use pandas to summary the dataset:

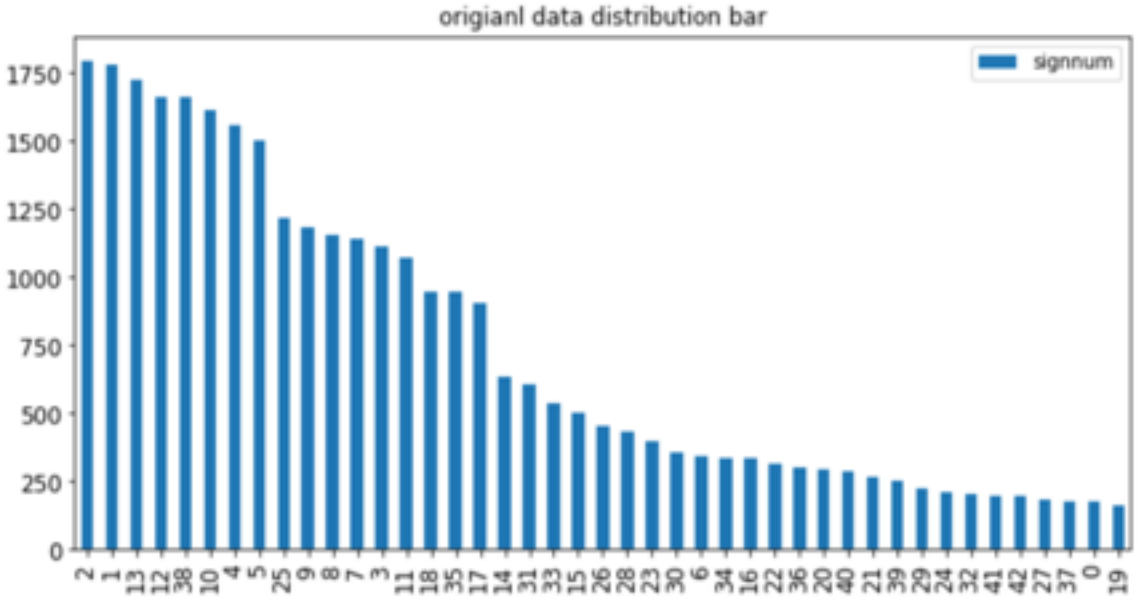


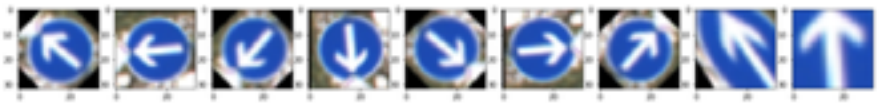
Summary:The training validation and test data is kind of has very similar distribution on different traffic sign, but the overall dataset is un-balanced, 50km/h traffic sign get 3000 samples, while 20km/h get only 500 below.

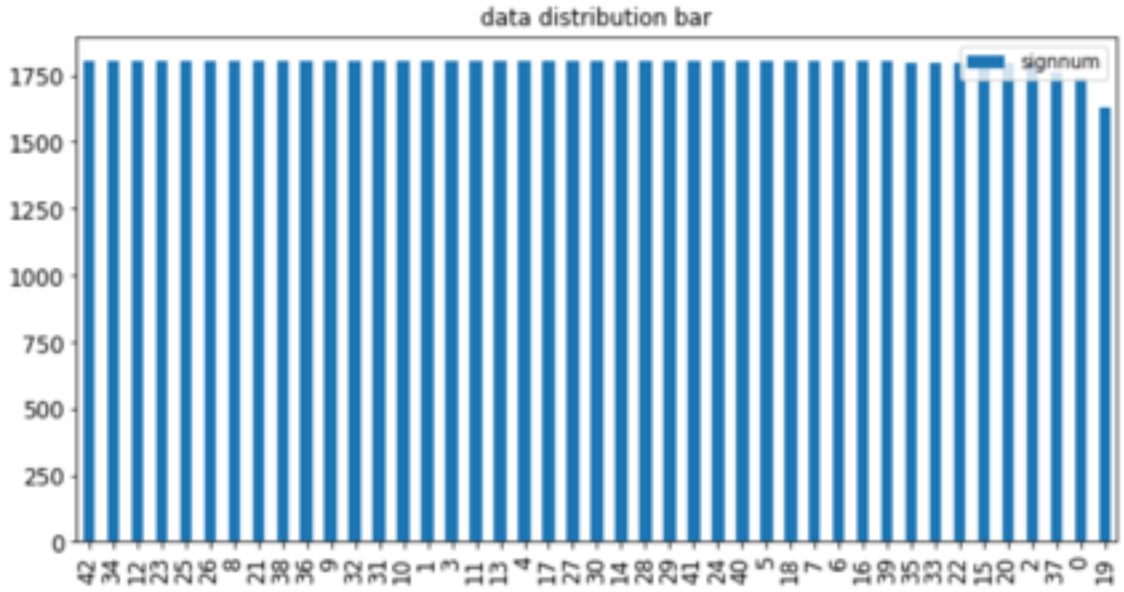
Later I found out the sign that has less sample is more likely to be predict as result, it make sense since it has less feature to match with since it has data sample to capture the feature from.

3. Augment the training data
    rotation,
    translation
    zoom
    flips
    color perturbation

```
(31367, 32, 32, 3)
```







The translation process like perspective, affine method, I use the same method for all picture and rotate 8 times for every picture which is not suitable for some picture, like the one which rotation has real meaning, and I got validation accuracy and test accuracy drop a little bit by augment training data.

grayscale, normalize

```
0.971(validation accuracy)
0.904(test accuracy)
0(accuracy on real image)
```

augment, no normalize

```
0.941
0.846
0.231
```

```
No augment, normalize
0.956
0.873
0.154
```

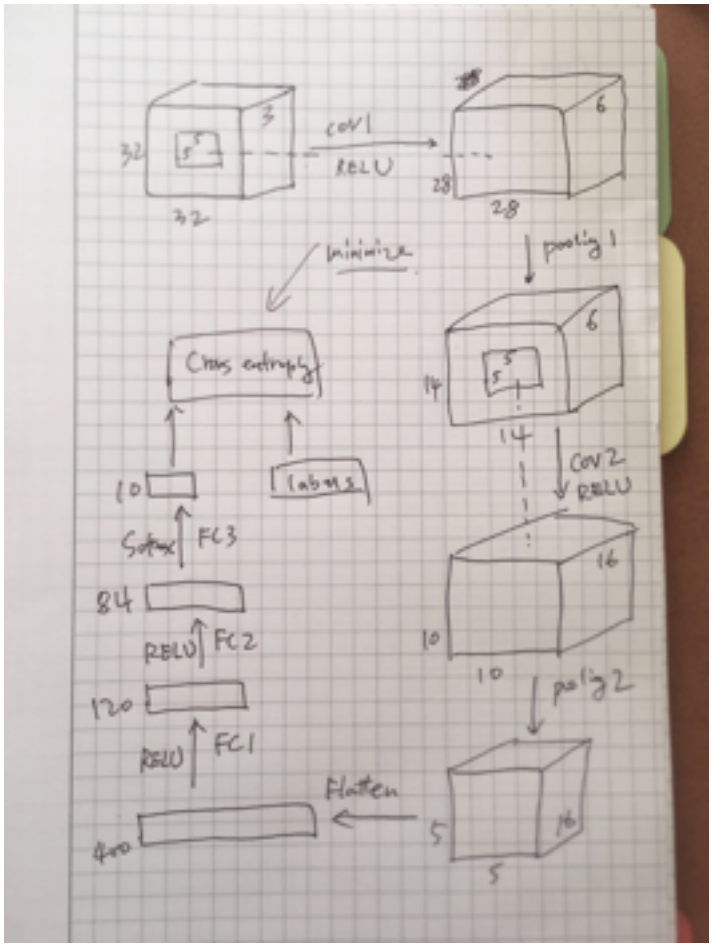
```
augment, normalize
```

0.965
0.857
0.231

Summary:
Grayscale help to increase the validation and test accuracy , but not on real accuracy, it overfit on the dataset.
Augment really help the real accuracy.
Normalize help at least not do harm in any situation

4. Model Architecture

2 conv, 2 polling, 3 fully-connected, 1 flatten and several activation node



At the very beginning, the init lenet was chosen and it give good result on training data /testing data from the same dataset, but not from captured data.

So I increase the depth of the network, so it can capture more feature, in conv, depth increase from 6 to 60, to 180, also it add a drop-out layer, and look like it increase a little bit accuracy on captured image.

It look like the data augment might be the key steps to increase the modue accuracy on captured data.

5. Model Training

```
logits = MyNet(x)
cross_entropy =
tf.nn.softmax_cross_entropy_with_logits(labels=one_hot_y, logits=logits)
loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate = rate)
training_operation = optimizer.minimize(loss_operation)
```

6. Solution Approach

Use Adam optimizer to minimize the cross entropy between predict and real labels of the label, and run 10 epoch with 128 as batch size and 0.001 as learning rate.

7. Acquiring New Images

Download 12 image from website.

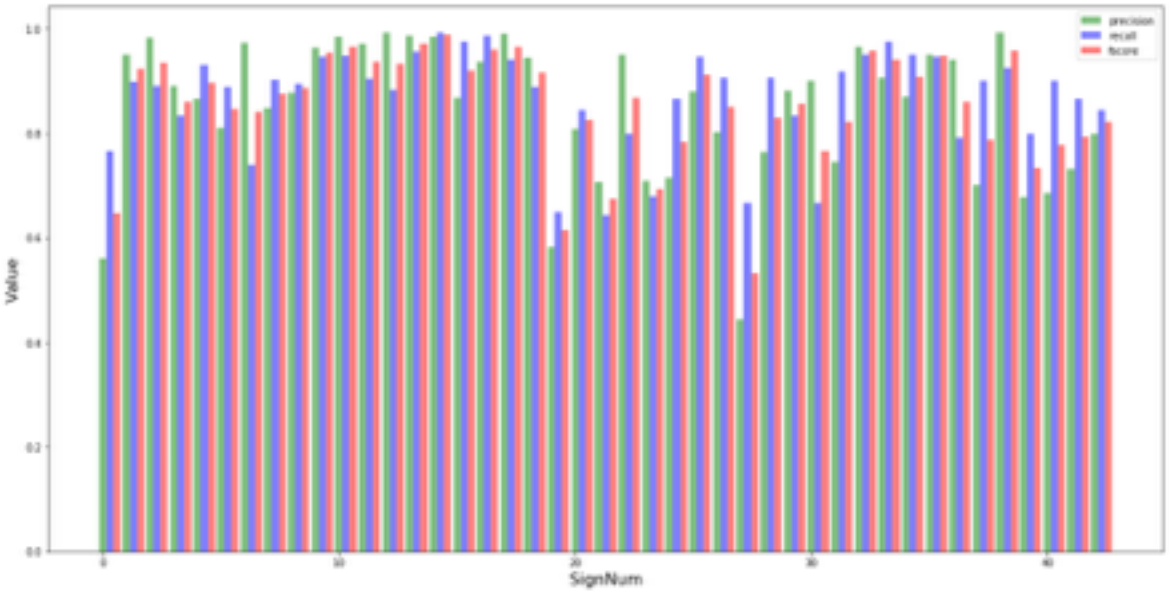
8. Performance on New real Image

The accuracy on the captured image is 23% while it was 90% on the testing set, thus it seem the model is overfitting.

Place to improvement: the color space and the photo clarity is quite different comping with dataset and real image, this could cause some error.Sometime the background and also perspective angle from the real

image can give the model a hard time. and can we add more layer so the model can recognize more feature?

9. Model Certainty - Softmax Probabilities



For most prediction, when it confidence level almost equal to 1, it most likely correct.

The 10 pic which supposed for 120km/h, but it predict as 20km/h, 20km/h sign has less sample than others, and it has low precision which can explain when it predict as 20km/h, it is quite unsure.