Capstone Project

2016 October 26th
Tomoaki Tsuzuki

Definition

Project Overview

In this project, the model that can decode sequences of digits from natural images are designed and trained. The model uses convolutional neural network for image recognition. The data set used is Street View House Number data which is available at this web site . The result is analyzed and future works are proposed.

Problem Statement

Detecting hand written or crafted multi digit numbers from real-world image or picture has number of applications such as Google street view housing number detection[*2]. However, it is usually hard for the computer to understand numbers from visual images. The problem is break down to 2 parts. In this project, methodology to detect multi digit number from images are discussed.

Metrics

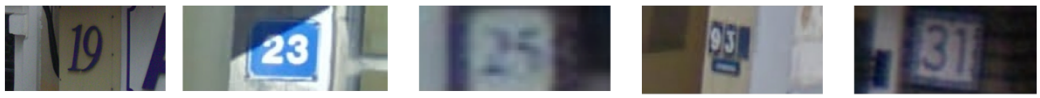
The metrics is fairly straight forward. After training the model, pictures with hand written multi digit number, which are not used to train the model, are fed to the model. The output is compared with the labels which those pictures. Numbers are correctly recognized only when all digits of numbers match with labels.

Analysis

Data Exploration

The data used is downloaded from here.

Below shows some of the data and labels downloaded.



19 23 25 93 31

Figure1. SVHN data

The picture has multi-digit numbers. The dimensions of picture are not organized and some of the numbers are fairly blur.

Below are the key statistical features of the data obtained.

	Training data	Test data
Number of data points	33402	13068

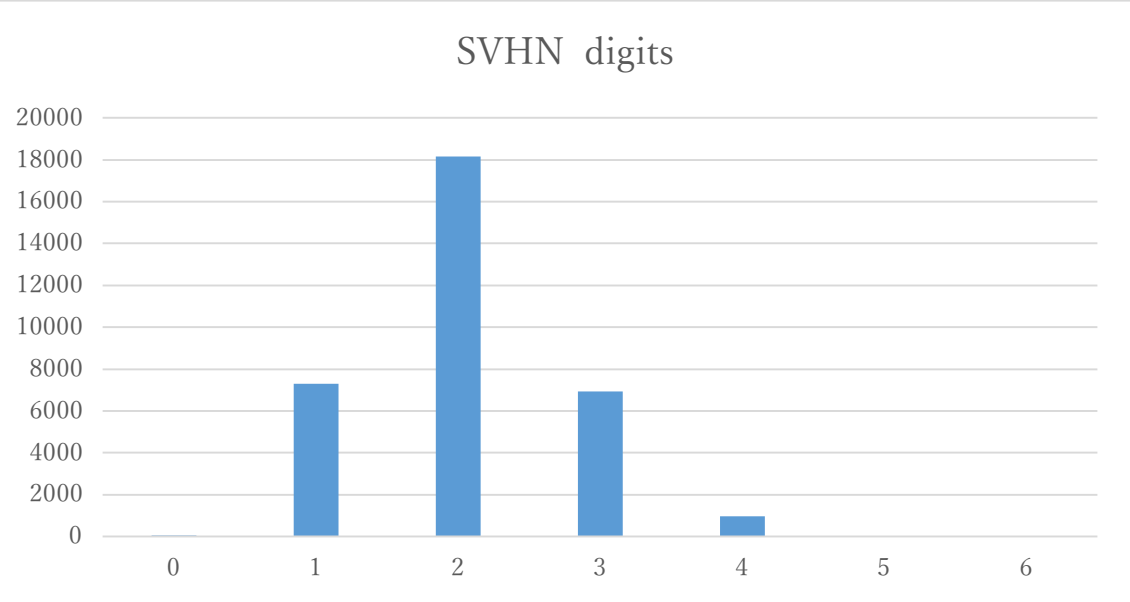


Figure2. Number of digits for all of the training data

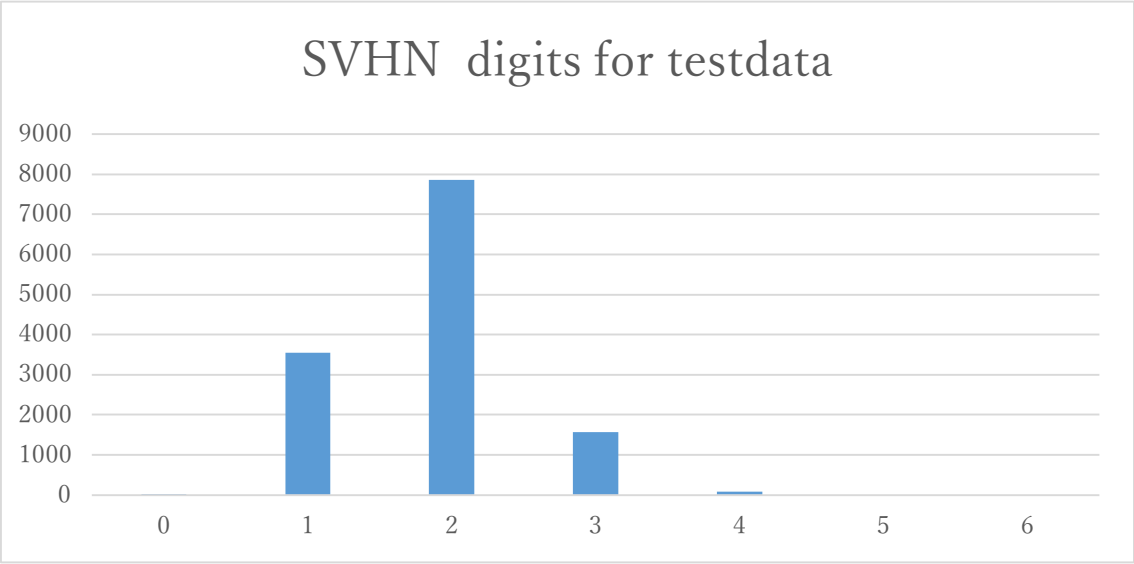


Figure3. Number of digits for all of test data

Exploratory visualization

The data comes with pre-defined information about box where all numbers reside inside of picture.

The image is sliced according to the box and those sliced images are used to train the model.

Below is the some of the extracted data.

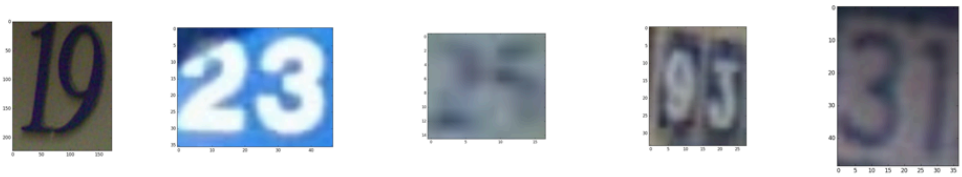


Figure4. sliced image with number

For models to be trained easily, all pixels in images are scaled such that it means to be 0 by below formula.

$$pixel = \frac{(pixel - 128)}{255}$$

Finally, image is resized to 32 * 32 pixels.
Some of the processed data looks like below.

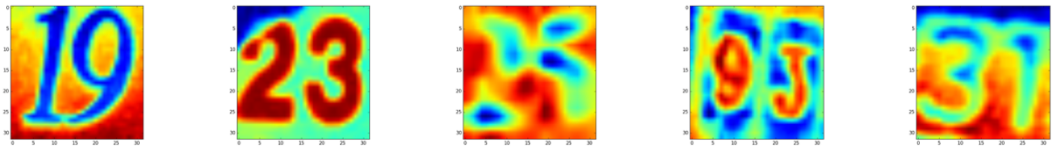


Figure5. processed data

Algorithms and Techniques

The algorithm used in this project is convolutional neural network which is suited for image recognition. Below is the overall architecture of the system.

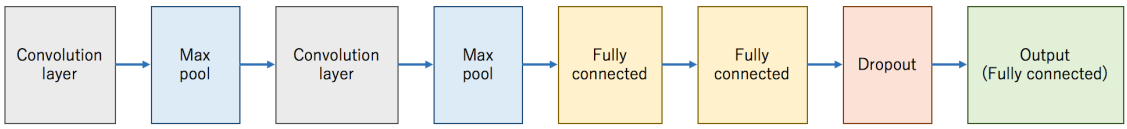


Figure6. Neural network architecture

The explanation of each layer is outlined below.

Why convolution?

In traditional neural networks, all pixel of data is input. If 10 * 10 pixels image, there are 100 input to the network.

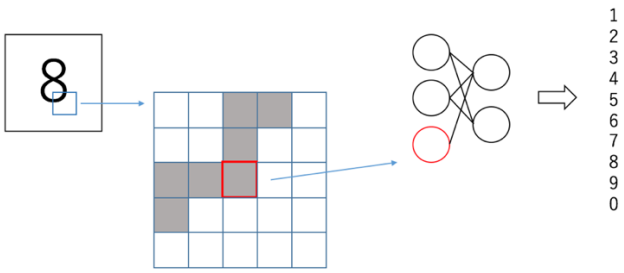


Figure7. Traditional Neural Net

This may work just fine depending on the application, but there could be weaknesses for the image which is slightly different from data used to train the model. Example of this scenario is using different fonts . This is illustrated by below picture.

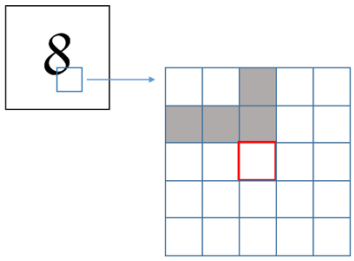


Figure8. Different font gives different input in pixel

The same number 8 is the input, but same exact input pixel has different value.

Overall shape of input is resembling so maybe we can improve the neural network performance by treating some area as input instead of 1 pixel. This data preprocessing idea is so called convolution.

What is convolution?

With convolution, small area (pre-fixed width and height) are taken from original picture and this area of information is “convoluted” as new data point.

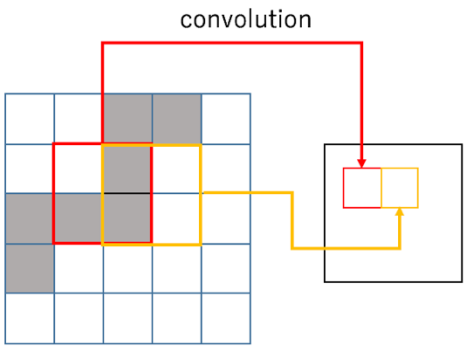


Figure9. Convolution

The process is iterated through all the pixels of the original data to create new data. Below picture shows how 5*5*3 filter create new data of 28*28*1 from 32*32*3 original image.

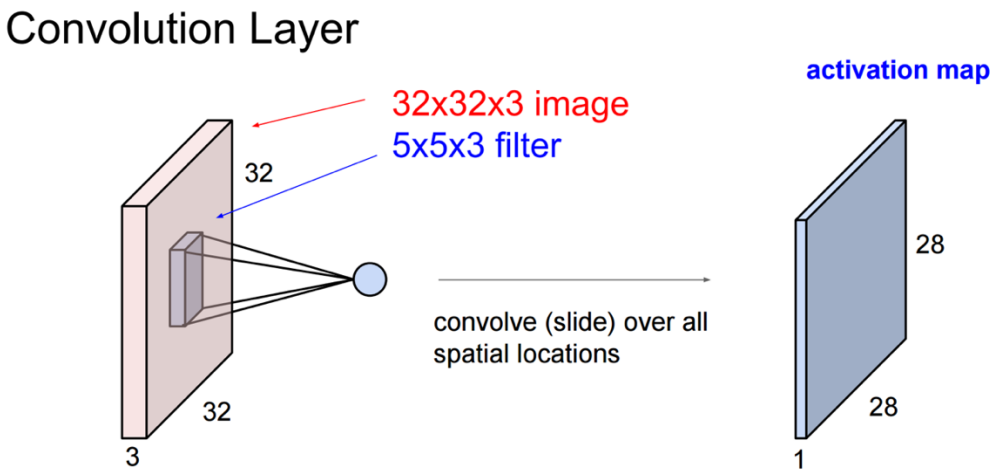


Figure10. Convolution illustration[3]

The created new data is then process through normal neural network for recognition of the image.

Pooling

Pooling is a function to reduce spatial size to reduce the amount of parameters and computation. Hence it is also to control overfitting. It is common to insert pooling layer in between convolutional layers. In this project, max pooling is used between convolutional layers which takes the maximum of all the data in given area. Example of 2*2 pooling is illustrated below.

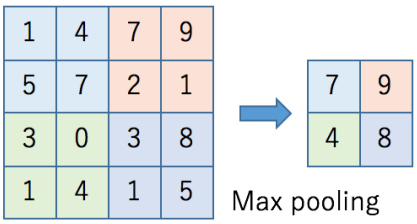


Figure11. Max pooling

Fully connected layer

Fully connected layer is traditional neural network layer.

Stochastic gradient descent

When training model, parameters of each layer are adjusted through iteration to get optimal performance. For doing this, loss of the model is defined as sum of differentiable equation. Gradient descent algorithm works as to minimize this loss at each iteration by taking derivative of loss function and adjust parameter to reduce loss. For not to move parameter too aggressively, it is common that learning rate is defined and only some portion of learning is reflected to next iteration. Stochastic gradient descent is the variant of gradient descent and it takes some portion of entire inputs and optimize parameters based on it. This will hugely reduce the burden of computation.

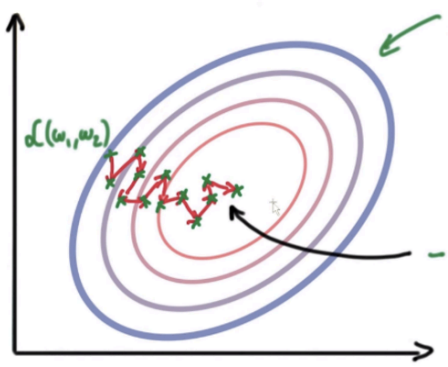


Figure12. Gradient descent[*4]

There are many optimizing algorithms and AdaGrad is one of them. It scales alpha learning rate based on history gradients.

Dropout

Dropout is the technique to suppress overfitting of the model. It randomly drops the output from layer. This helps the model to be more generic and not to be trained for the features which are not relevant to the prediction.

Benchmark

The benchmark of this project is Google's paper[*5].

It states that the accuracy of their system for recognizing street view house number from images taken from real world is over 90%. So I hope my approach will get to somewhat close to that accuracy.

Methodology

Data Preprocessing

Data preprocessing has been done as below procedure.

1. Download data set from the internet
2. Slice the original image to the one containing multi digit numbers
    2-1. Slice is done by the outline that comes with the image
3. Image color is averaged so that image has only one layer of color
4. Image is resized to 32*32 pixels
5. Each pixel is scaled such that means to be 0

This processed data is used to train and test model accuracy.

Implementation

Tensorflow (https://www.tensorflow.org/) is used to implement convolutional neural network. In this project, I have used below as convolutional layers as well as fully connected layer and output layer.

- 8*8*5 convolutional layer
- max pooling
- 4*4*55 convolutional layer
- max pooling
- fully connected layer
- fully connected layer
- dropout (20% is dropped)
- output layer

Output layer has 5 weights and biases for take into account multi-digits recognition.

```
def model(data):
    conv = tf.nn.conv2d(data, layer1_weights, [1, 1, 1, 1], padding='SAME')
    pool = tf.nn.max_pool(conv, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    hidden = tf.nn.relu(pool + layer1_biases)
    conv = tf.nn.conv2d(hidden, layer2_weights, [1, 1, 1, 1], padding='SAME')
    pool = tf.nn.max_pool(conv, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    hidden = tf.nn.relu(pool + layer2_biases)
    shape = hidden.get_shape().as_list()
    reshape = tf.reshape(hidden, [shape[0], shape[1] * shape[2] * shape[3]])
    hidden = tf.nn.relu(tf.matmul(reshape, layer3_weights) + layer3_biases)

    logits1 = tf.matmul(hidden, s1_weights) + s1_biases
    logits2 = tf.matmul(hidden, s2_weights) + s2_biases
    logits3 = tf.matmul(hidden, s3_weights) + s3_biases
    logits4 = tf.matmul(hidden, s4_weights) + s4_biases
    logits5 = tf.matmul(hidden, s5_weights) + s5_biases
    return [logits1, logits2, logits3, logits4, logits5]
```

There are 5 logits output. Those are softmaxed and cross entropy is calculated and averaged. The loss function is defined as sum of those.

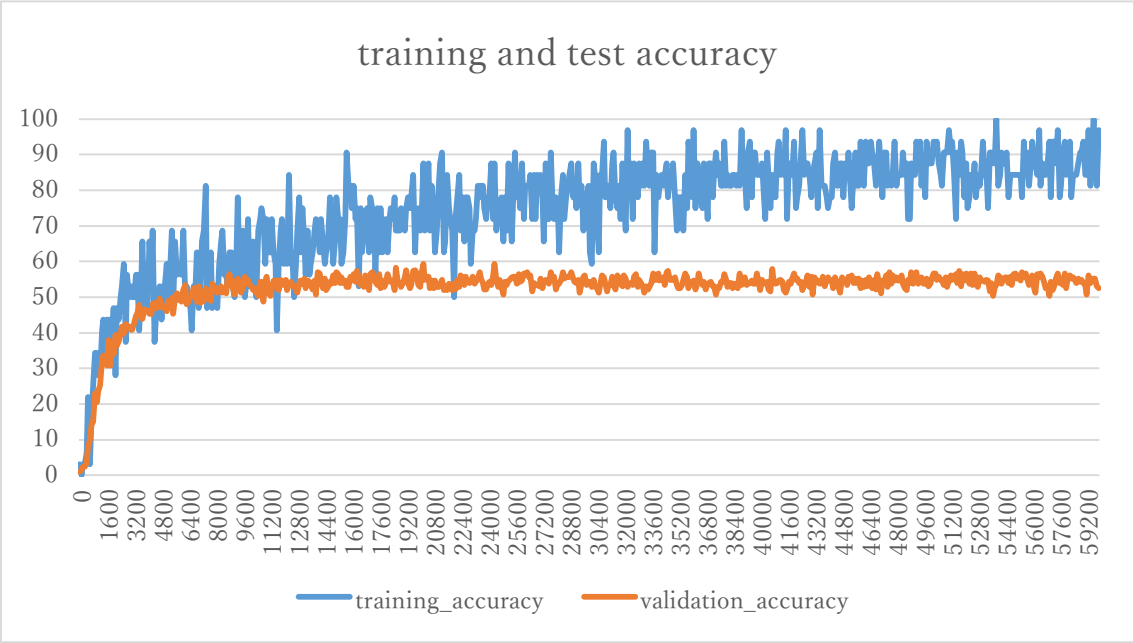
```
logits = model(tf_train_data_set)
loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits[0], tf_train_labels[:, 1])) +¥
      tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits[1], tf_train_labels[:, 2])) +¥
      tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits[2], tf_train_labels[:, 3])) +¥
      tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits[3], tf_train_labels[:, 4])) +¥
      tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits[4], tf_train_labels[:, 5]))
```

For optimizer, Adagrad optimizer is used.

Result

Model evaluation

Below is the prediction accuracy for training data set and validation data set. The training is done 60,000 times. Accuracy is calculated as correct prediction divided by number of data times 100. Data is regarded as correctly predicted when all digit matches exactly.



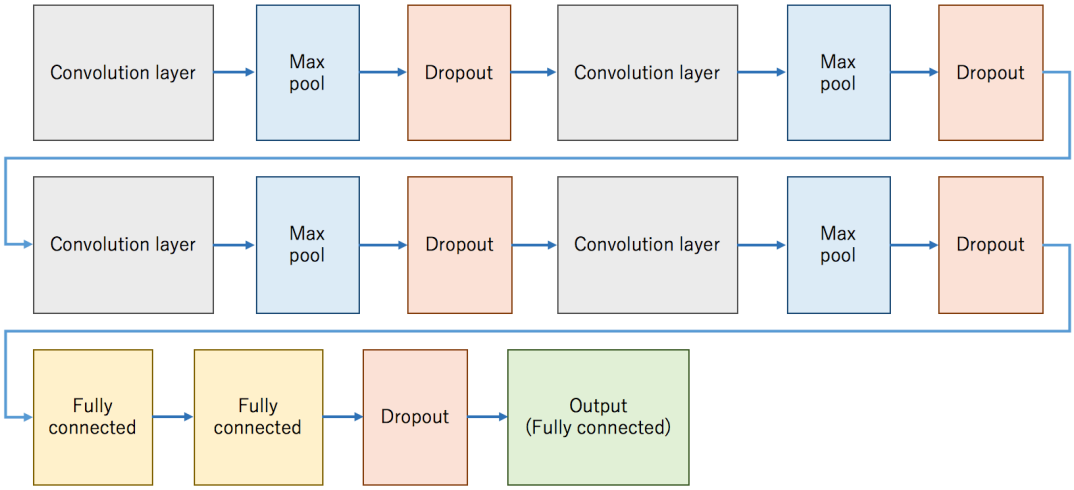
As it can be seen, training accuracy increases towards 100% whereas validation accuracy stays about 50%.
This is clearly the sign of overfitting.

Improvements

To overcome the overfitting problem, below are tried.

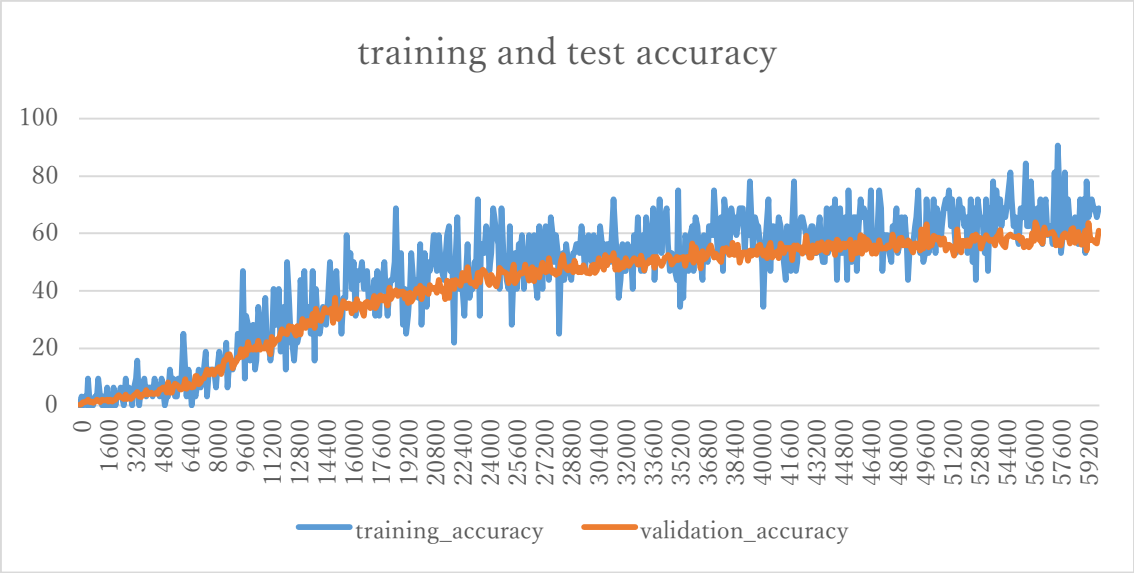
- Deeper network (2 additional convolutional layer)
- Dropout at each convolutional layer
- Regularization for loss function

Modified architecture is as below.



- 8*8*5 convolutional layer
- max pooling
- dropout (20% is dropped)
- 4*4*55 convolutional layer
- max pooling
- dropout (20% is dropped)
- 2*2*110 convolutional layer
- max pooling
- dropout (20% is dropped)
- 1*1*220 convolutional layer
- max pooling
- dropout (20% is dropped)
- fully connected layer
- fully connected layer
- dropout (20% is dropped)
- output layer

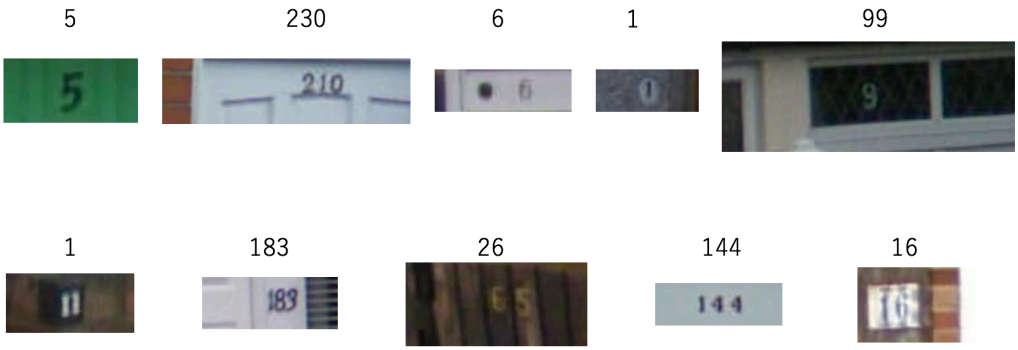
Below are the accuracy of training and validation data.



Accuracy of validation get close to 60% so there are some improvements. However, it is not in the level of being used in real world application.

Justification

Below pictures are some of the test data and predicted numbers with the trained model.



As in the validation accuracy, 60% of the image is predicted correctly.

Conclusion

The convolutional neural network model is trained for recognizing multi digit numbers from pictures. While model predict multi digit numbers from real world images, it should be improved to be used in the real world applications.

To improve, something like below should be more investigated.

- More advanced architecture (Like Google's inception model)
- Collect more training data or create more data by tweaking existing ones
- More experiments and adjustments of the hyper parameters

References

*1 Udacity Deep learning course: https://www.udacity.com/course/deep-learning--ud730

*2 The Street View House Numbers Dataset: http://ufldl.stanford.edu/housenumbers/

*3 CS231n: Convolutional Neural Networks for Visual Recognition Slide: http://cs231n.stanford.edu/slides/winter1516_lecture7.pdf

*4 Deeplearning4j: https://deeplearning4j.org/updater

*5 Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks: http://static.googleusercontent.com/media/research.google.com/ja//pubs/archive/42241.pdf

*5 Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks https://arxiv.org/pdf/1312.6082.pdf