Capstone Project

2016 October 27th
Tomoaki Tsuzuki

Definition

Project Overview

With the advent of information technologies, there are more and more digital image becomes available. Detecting numbers and characters from those images are called optical character recognition (OCR) and becomes important in the field of computer vision. Optical character recognition has been studied since early 20th century and it has numbers of applications such as text-to-speech from visual image.

In this project, multi digit number detection from digital image is studied and implemented. This problem is addressed by Google to detect house numbers from street view images[*1]. This project describes theory behind multi digit number recognition and implementation is proposed. Result is analyzed and compare with the result from the paper.

Problem Statement

Detection of numbers from image is supervised classification problem. The images with labels (correct numbers in the image) is prepared as training dataset.
Neural network model which makes classification is trained with the training dataset. Once model is trained, any images with number can be fed to the model. The model determines digits inside the image (if any) and output it.
In this project, preparation of dataset, training model, and making prediction are addressed.

Metrics

The metrics used in this project are

- Accuracy
- Precision
- Recall
- Training duration

Accuracy is calculated by counting number of correct prediction divided by total number of dataset. There is no partial score and all digits must be right to be counted as correct prediction.

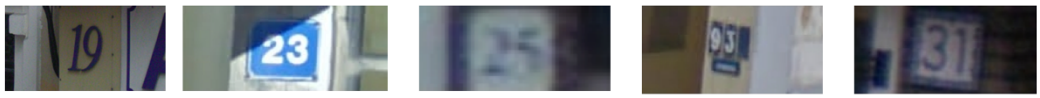
Precision and recall are selected as metric because it shows classifier performance quantitatively. Precision is a measure of the accuracy provided that a specific class has been predicted. Recall is a measure of the ability of a model to select instances of a certain class from a data set.
In our case, precision and recall are calculated for each labels and averaged to get overall picture of what is the accuracy of the model.

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
Number of label types	1840	757

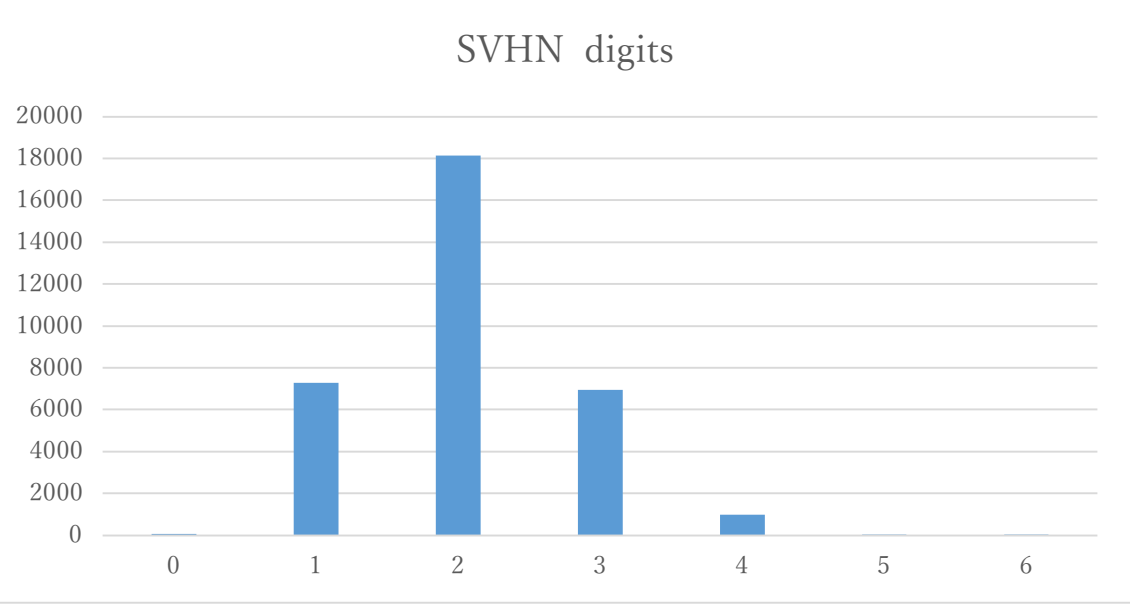


Figure2. Number of digits for all of the training data

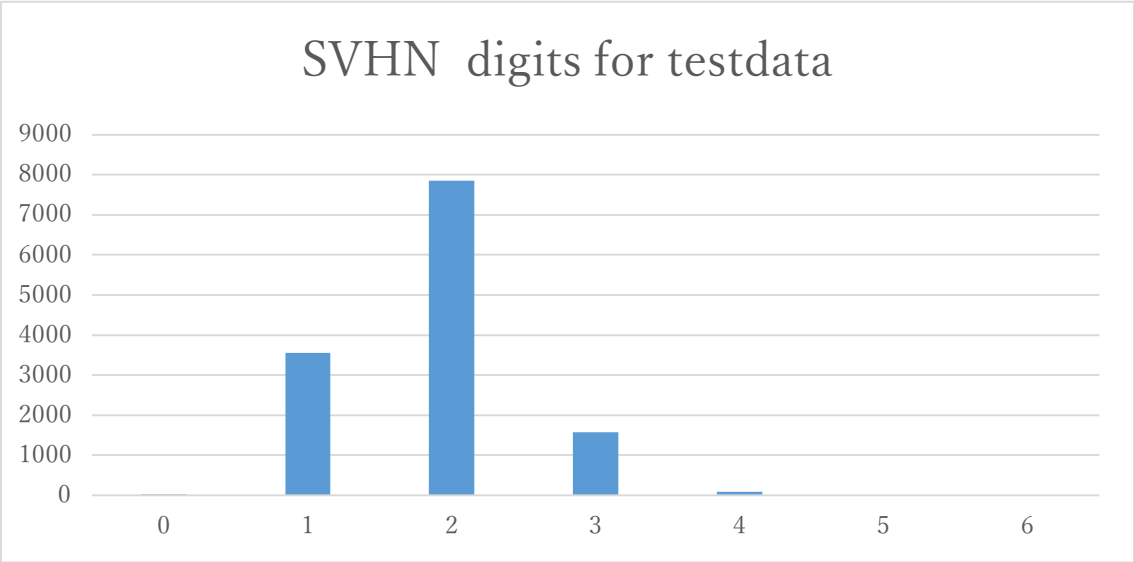


Figure3. Number of digits for all of test data

Exploratory visualization

The images are not really good quality as it is the real world picture. Some of the attributes are

- Some of the picture is blur
- Some numbers are rotated
- Size of the image are not consistent
- Where the numbers are found in the pictures is not consistent
- Some of the numbers in pictures are too dark or light

The data is pre-processed prior to be used.

To get only numbers from picture, the pre-defined information about coordinates where all numbers reside in the picture is used. This information comes with the dataset. The image is sliced according to the box. Below is the some of the extracted data.

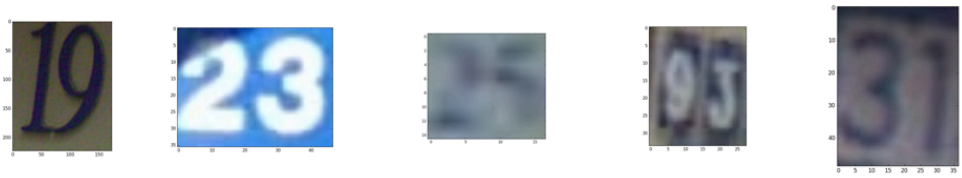


Figure4. sliced image with number

Although there are 3 color channels, the all color information is averaged to get grayscale picture. And all pixels in images are scaled such that it means to be 0 by bellow formula.

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

This preprocess is hopefully help neural networks understand numbers more easily and reduce effect of blurriness and lighting issue.

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

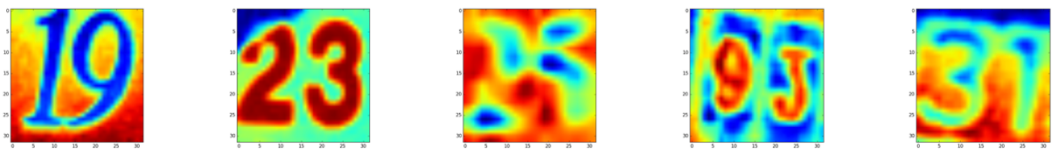


Figure5. processed data

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.

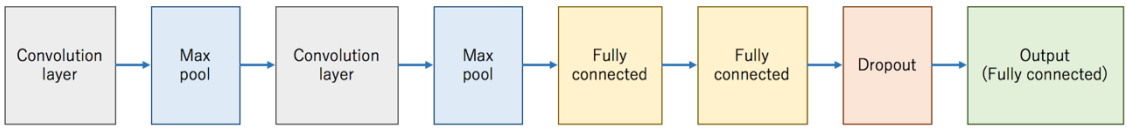


Figure6. Neural network architecture

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.

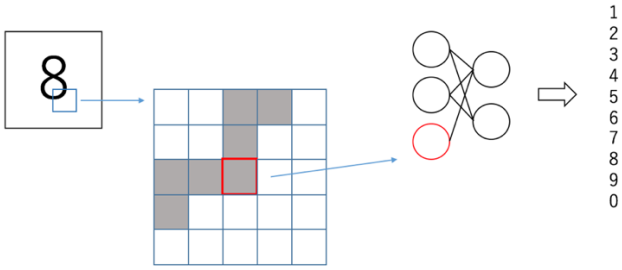


Figure7. Traditional Neural Net

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.

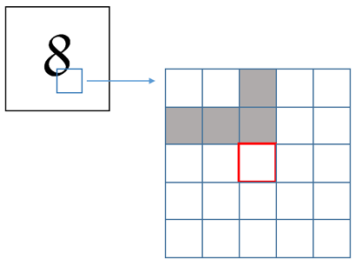


Figure8. Different font gives different input in pixel

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.

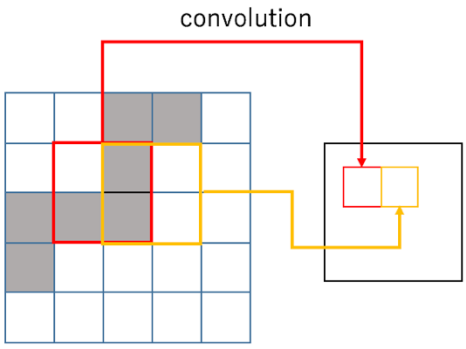


Figure9. Convolution

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.

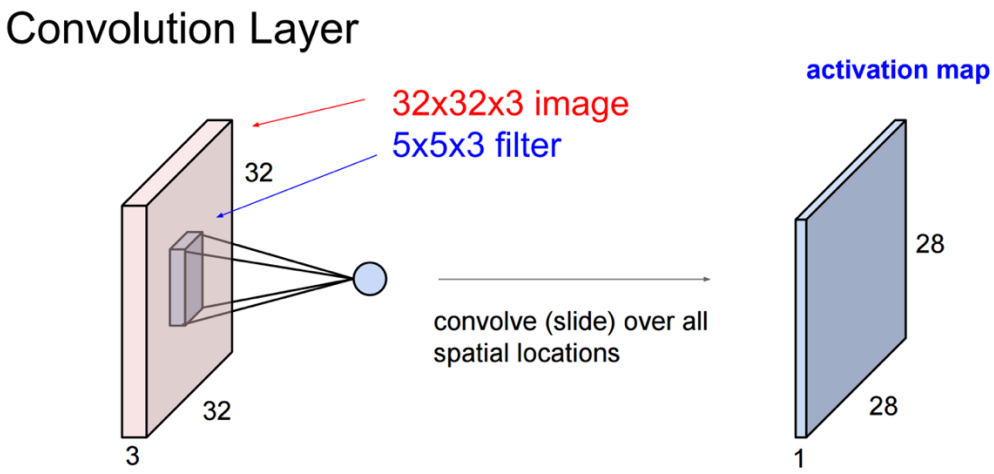


Figure10. Convolution illustration[*3]

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.

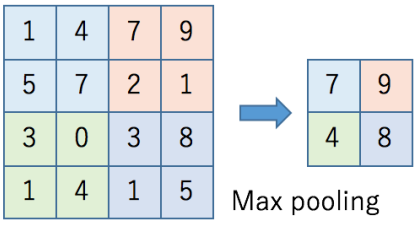


Figure11. Max pooling

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.

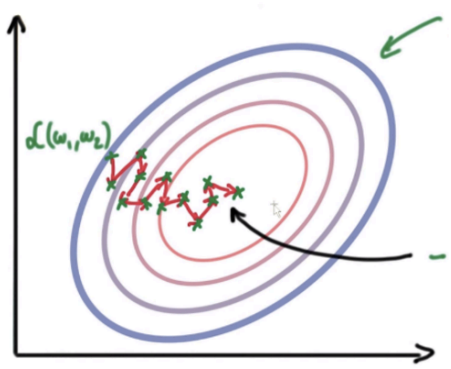


Figure12. Gradient descent[*4]

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 model of this project is the one described in Google's paper[*1].

In the paper, it is stated that their best model has 98% accuracy. The time required to train this model is about 6 days using 10 replicas in large scale distributed deep networks[*6].
Their model consists of 8 convolutional hidden layers, one locally connected hidden layer, and 2 densely connected hidden layers.

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

The coding is divided into 2 major class implementation, SVHN class and CNN class.

SVHN class

SVHN class is the data container. It has below functionality.

- Prepare training and test data
- Provide some functions to analyze and visualize data

Public methods available from this class are below.

Name	Description
__init__	Constructor When created, the instance automatically prepares training and test data. If data is not found in local machine, it will automatically get it from internet.
show_as_image	Show first n (default 10) images from training dataset

get_histogram_data	Get training data to plot histogram
get_test_histogram_data	Get test data to plot historgam
get_reformatted_dataset	Get training dataset which can be fed to CNN class
get_reformatted_test_dataset	Get test dataset which can be fed to CNN class

CNN Class

CNN class is the neural network model. It has below functionality

- Train model with training dataset (images and labels)
- Prediction of the numbers frm image

Public methods available from this class are below.

Name	Description
__init__	Constructor You can set parameters if needed (all of the parameters have default value)
fit	Train the model with training data set
test	Test the model with test data set
predict	Predict numbers from image

Tensorflow (https://www.tensorflow.org/) is used to implement convolutional neural network in CNN class. 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.

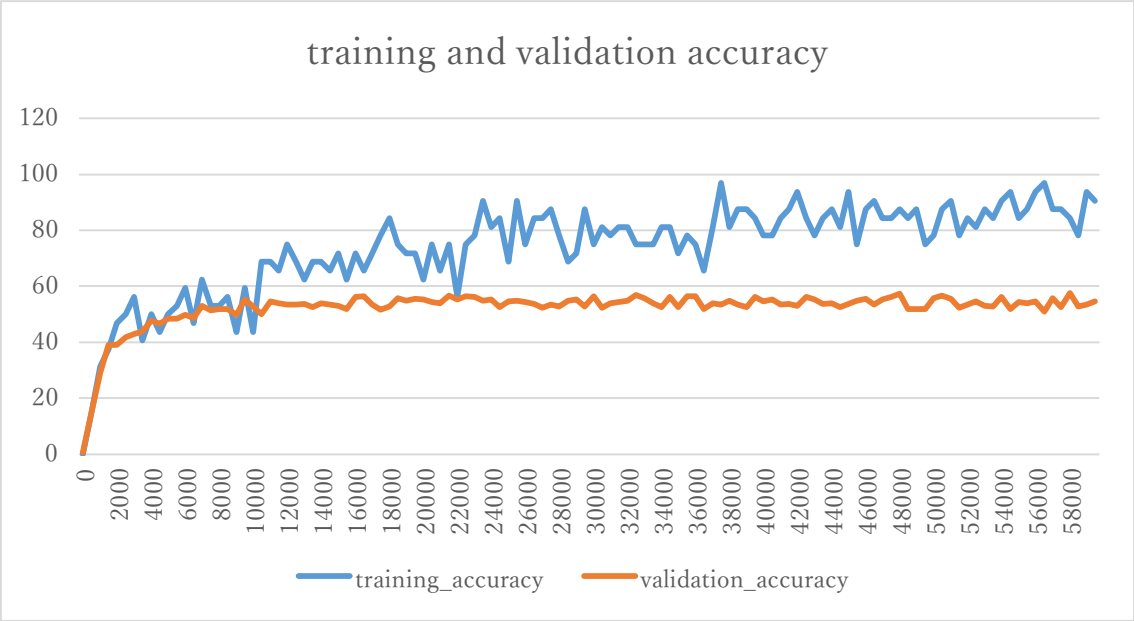
There are 5 logits output. Those are softmaxed and cross entropy is calculated and averaged. The loss function is defined as sum of those.

For optimizer, Adagrad optimizer is used. It adapts the learning rate to the parameters, performing larger update for infrequent and smaller updates for frequent parameter. For this reason, it is suited for sparse data and I have decided to use this optimizer.

Result

Model evaluation

Below is the prediction accuracy for training data set and validation data set.
The training is done 60,000 times.



Below summarizes the key metric for this model.

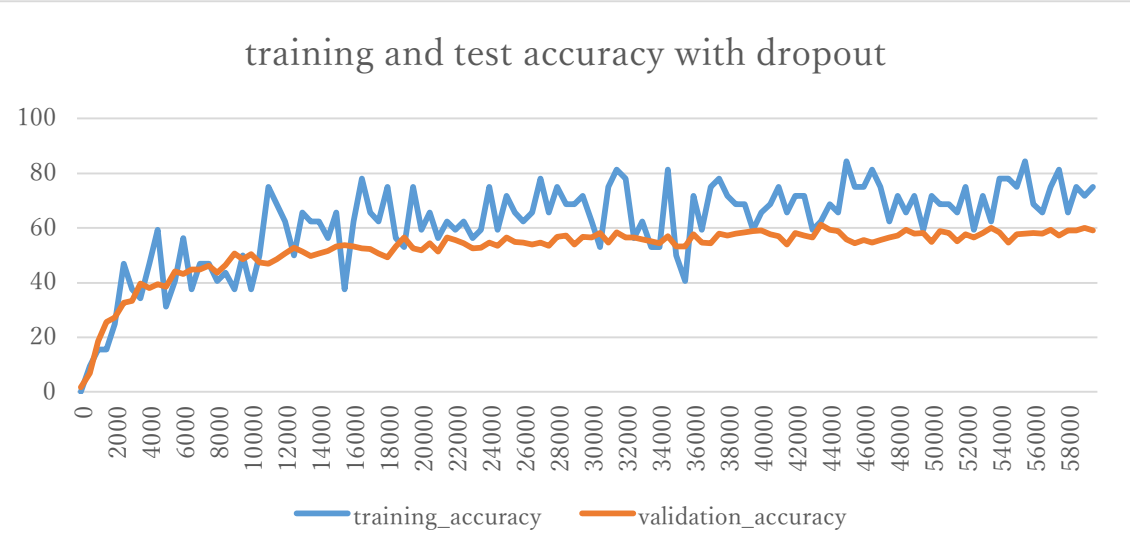
Parameter	Value	Unit
Time to train	2215	sec
Test accuracy	59.3	%
Precision	0.6	
Recall	0.6	

As it can be seen, training accuracy increases towards about 90% whereas validation accuracy stays about 55%. This is clearly the sign of overfitting.
It is also clear that test accuracy is about the same as validation accuracy.

Improvements

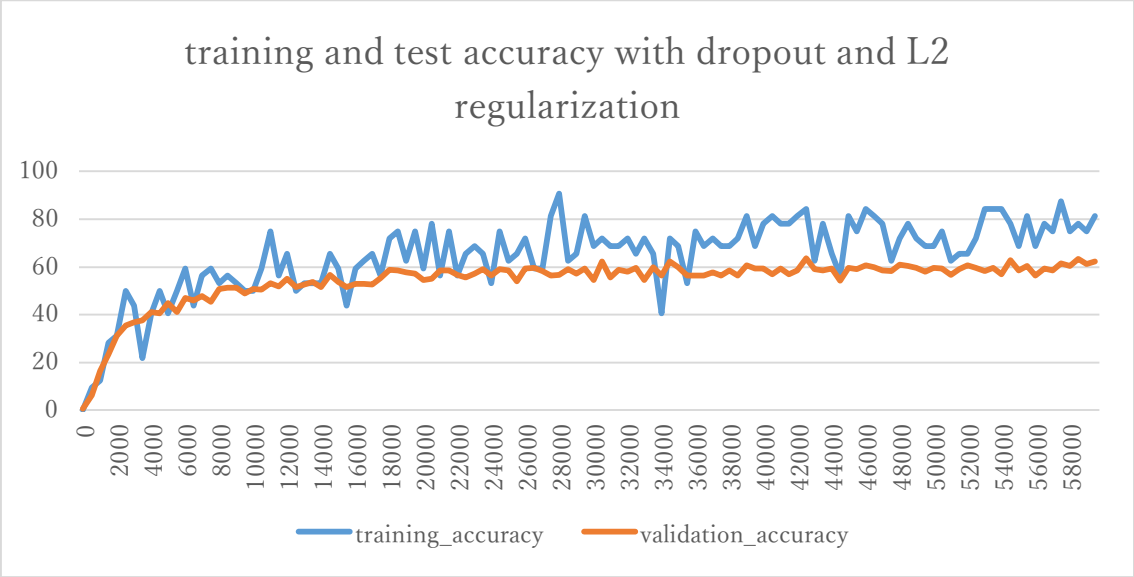
To overcome overfitting, dropout is included at each convolutional layer. Dropout ratio is set to 0.8 for all layers which means 20% of outputs are dropped.

Below are the accuracy and key metrics for this setting.



Parameter	Value	Unit
Time to train	2178	sec
Test accuracy	61.9	%
Precision	0.6	
Recall	0.6	

The performance is slightly better than the previous setting. However, there still is signs of overfitting. To further reduce the effect of overfitting, the L2 regularization is introduced. Below is the result.

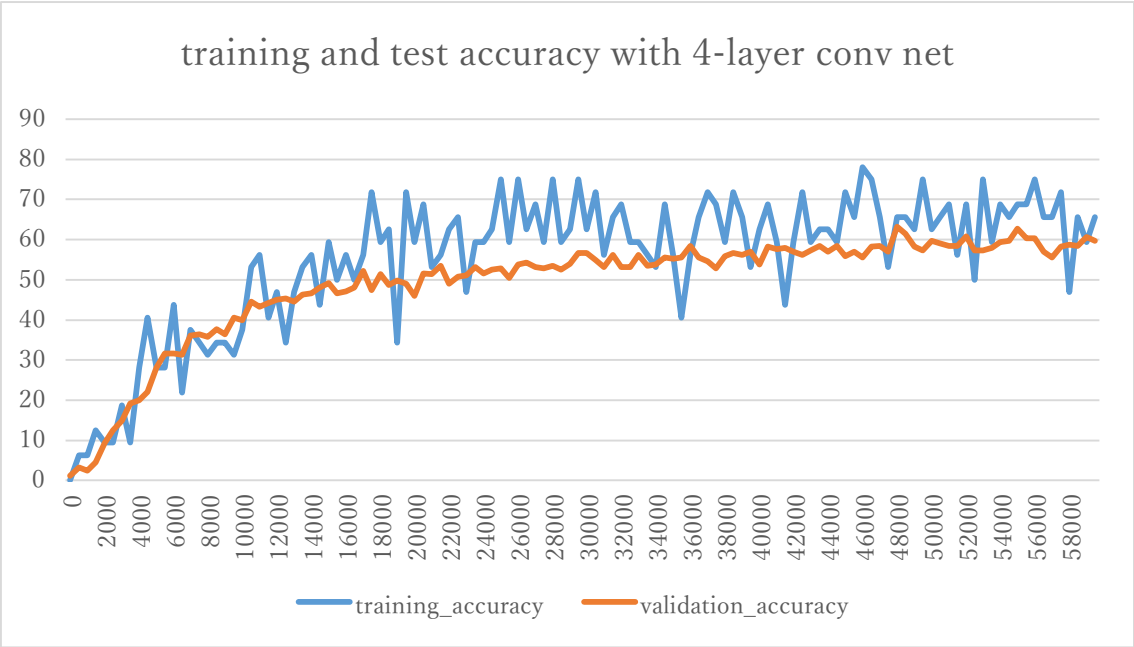


Parameter	Value	Unit
Time to train	2335	sec
Test accuracy	63.5	%
Precision	0.6	
Recall	0.6	

Again, there is slight increase in test accuracy, but overall the result did not improve much.

I have then tried to use deeper network. Instead of 2 layers, 4 layers are used.

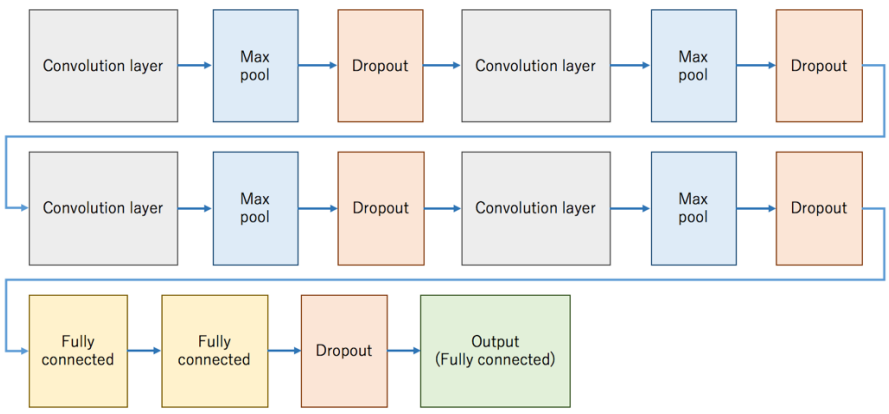
- 8*8*5 convolutional layer
- 4*4*55 convolutional layer
- 2*2*110 convolutional layer
- 1*1*220 convolutional layer



Parameter	Value	Unit
Time to train	3098	sec
Test accuracy	65.4	%
Precision	0.7	
Recall	0.7	

Again, the result is slightly better than previous experiment. It seems having more layer helps to improve performance.

The final model used looks like below.

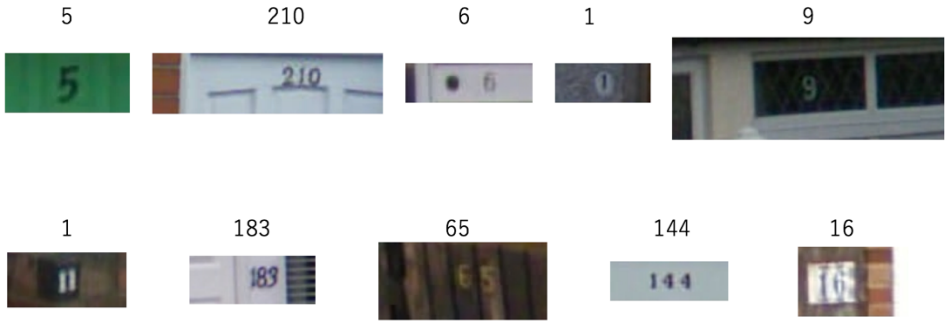


- 8*8*5 convolutional layer
- max pooling
- dropout (20% is dropped)
- 4*4*55 convolutional layer
- max pooling
- dropout (20% is dropped)
- 2*2*110 convolutional layer
- max pooling
- dropout (20% is dropped)
- 1*1*220 convolutional layer
- max pooling
- dropout (20% is dropped)
- fully connected layer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

Comparing this result with benchmark model, it is clearly worse performance as the benchmark states it has 98% accuracy. The difference may be because of the fact that model layers are shallower. The benchmark model has 8 convolutional layers whereas ours has 4 layers. Because of limited computational resource available, I could not train the model in such a large scale in this project.

Justification

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



For this set of data, 90% of the numbers are predicted correctly.

Since number of labels types are too large (757) in test data, it is hard to analyze something like confusion matrix. To find out model strangeness and weakness, the accuracy for each digits are analyzed.

	Predicted Correctly	Accuracy
1 digit number	1986	81%
2 digits number	5715	68%
3 digits number	869	41%
4 digits number	1	0.006%
5 digits number	0	0%

It is clear that the model has good performance to predict single digit number, and as digits goes high, prediction performance gets lower.

Conclusion

The convolutional neural network model is trained for recognizing multi digit numbers from pictures. While model predict multi single digit numbers from real world images in reasonable accuracy, it has weakness to predict higher order numbers correctly.

Since having more layers improve accuracy, it would probably increase performance of the model with higher order convolutional layer, although that requires lots of computational power. Since the model has better accuracy for predicting single digit number, it would be good to further study different approach such as scanning through entire images just looking for single digit number and construct multi digit number with the result.

Another point to note is that training neural network requires a lot of experiments and time to train the model. It is quite time consuming to find out what are the key parameters and good values. To train deeper and deeper networks, one should really consider preparing scalable computational environment before diving into training.

References

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

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

*6 Large Scale Distributed Deep Networks https://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf