# Udacity Deep Learning - Final Project

# The Street View House Numbers (SVHN) multiple numbers recognition

## Problem Define:

We want machine learn how to read number strings in the real world. To achieve this goal, we need a more realistic dataset, so we choose SVHN, which contains the house numbers in Google Street View, as our main dataset (Example show as Figure 1.).

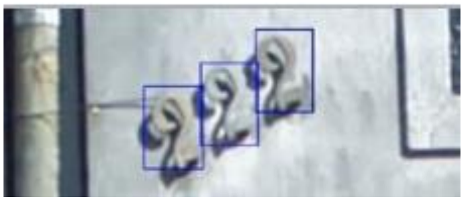
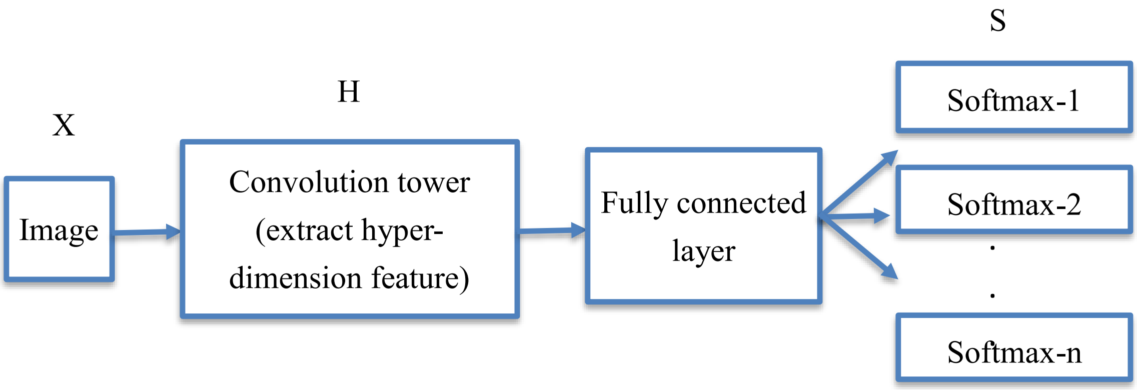


Figure 1. An illustration of data in SVHN dataset

## Model:



(reference from *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks*)

We want to machine can read the number string S; it means that we need P(S|X) as higher as possible. Due to we can treat each digit in string is independent, so we can derive P(S|X) to Equation (1).

$P(S|X) = P(S_1|X) \times P(S_2|X) \times .... \times P(S_n|X)$ Equation (1)

Where:
X: The pixel array of any image
S: the number string, which can be split to independent variables $S_1$, $S_2$, …. $S_n$ (n is an arbitrary number determined at the beginning)

We also introduced convolutional layer into our network, so that it can extract some useful regional information. Based on this information, we were capable to distinguish between digits with same value but different location. After adding convolutional layer, we could modify Equation (1) to Equation (2).

$$P(S|X) = P(S|H) = P(S_1|H) \times P(S_2|H) \times .... \times P(S_n|H) \qquad \text{Equation (2)}$$

Where:
H: output of convolutional layer (hyper-dimension features)

## Implementation:

Data Overview:
There are three datasets on SVHN website: train, test and extra dataset. Through simple data analysis, we could find the distribution of the length of number string, and the result is shown as Figure 2.

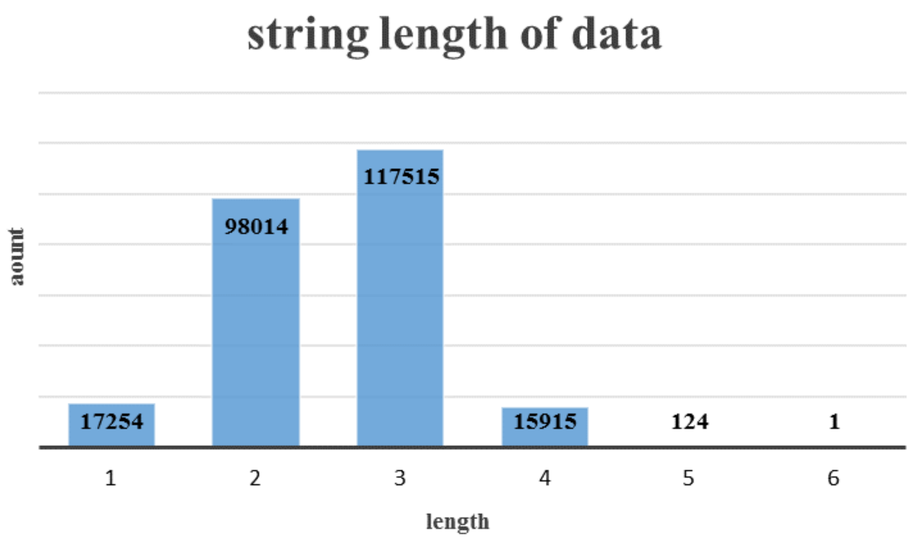


Figure 2. string length of data (train dataset and test dataset)

According to Figure 2., there is only one string has length 6. To decrease the amount of parameters while not harm the performance to much, so I choose 5 to be the maximum length of string in the program.

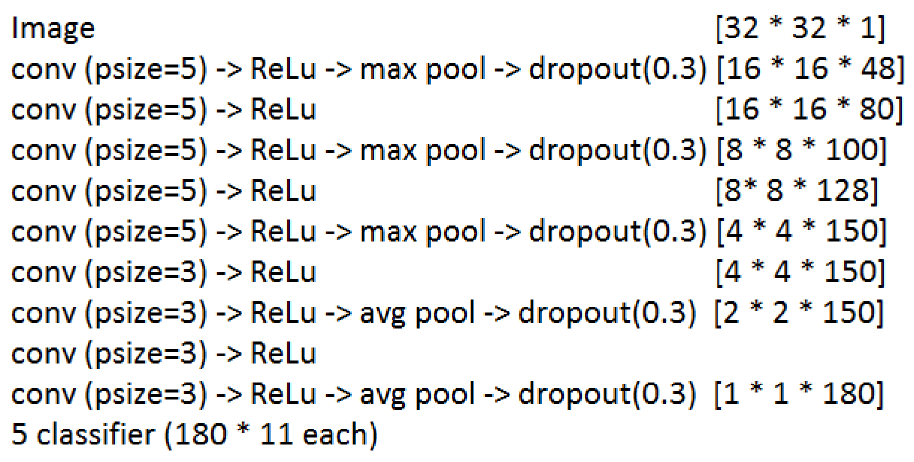
I split the data into 3 part: train, validation and test set which size are about 230000, 5000 and 5000, separately. (The reason to why the size of validation and test set is small: If the size of validation or test set over 8000, I will run out of the memory…)

Data Preprocessing:
SVHN is a realistic dataset, but it offers the location of every digit. After founding the bounding box containing the string, I crop the image in the bounding box and resize it to 32 × 32 pixels. This preprocessing is done for all datasets.

Multi-classifier:
There are 5 output classifiers in the network, and each has 11 classes: 0~9 means character 0~9; 10 means not a number. So if one label is 153, then the correct outputs of classifiers are 1, 5, 3, 10, 10.

Loss Function:
The loss is sum of cross entropy of all classifiers. If we minimize the loss, our prediction will be more close to the labels, P(S|H) becoming higher, which is our goal.

Network Architecture:
I build a similar network with the paper, but do a little modification (reference to *https://github.com/znat/udacity-digit-recognition-program-svhn/blob/master/project-report/project-report.md*). Detail of architecture is shown in Figure 3.



```
Image                                                    [32 * 32 * 1]
conv (psize=5) -> ReLu -> max pool -> dropout(0.3) [16 * 16 * 48]
conv (psize=5) -> ReLu                                   [16 * 16 * 80]
conv (psize=5) -> ReLu -> max pool -> dropout(0.3) [8 * 8 * 100]
conv (psize=5) -> ReLu                                   [8* 8 * 128]
conv (psize=5) -> ReLu -> max pool -> dropout(0.3) [4 * 4 * 150]
conv (psize=3) -> ReLu                                   [4 * 4 * 150]
conv (psize=3) -> ReLu -> avg pool -> dropout(0.3)  [2 * 2 * 150]
conv (psize=3) -> ReLu
conv (psize=3) -> ReLu -> avg pool -> dropout(0.3)  [1 * 1 * 180]
5 classifier (180 * 11 each)
```

Figure 3. information of the architecture of my neural network

The structure of my network is not very deep, but I still apply some technique to reduce the risk of gradient vanishing. I use Variance-Scaling-Initializer to initial all the weights (no on biases). I even try to take batch normalization to all layers or use scaled exponential linear units (SELU) as activated function, but they only decreased the accuracy of validation a lot, so I didn't add them to the final architecture.

## Result:

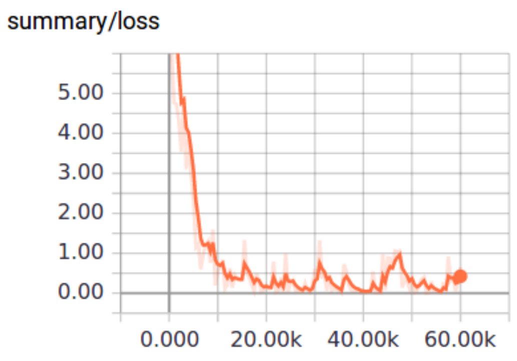
Learning rate: 0.05 with exponential decay 0.95 every 10000 steps

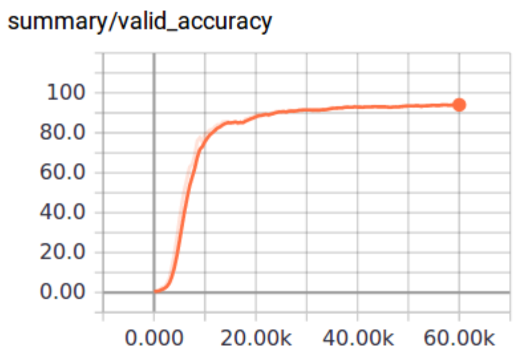
Iteration: 60000

Batch size: 16

Length of string: 5

Training time: 16 hours



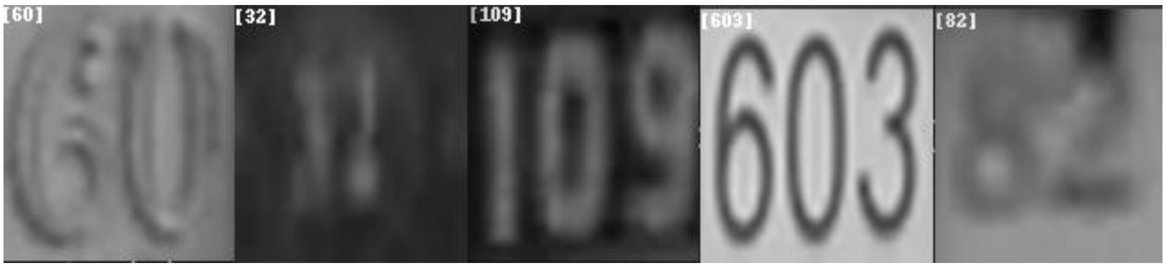


Final result:

Validation: 94.1%

Test: 94.5%

Part of the recognition result in test set:



Some of the number strings are so blur that even human might hardly recognize them, but the machine could still give the correct answers, and I am satisfied to this result.

## Conclusion:

After finishing this project, I have more familiar with Tensorflow and build machine learning model with it.
For SVHN dataset, the best accuracy already announced is 98.3%, which might beat human's eyes, so I think there is not much room to improve for this problem. However, SVHN is still a good material for beginner to learn to use deep learning to solve problem.

## Reference

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*

Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks

https://github.com/znat/udacity-digit-recognition-program-svhn/blob/master/project-report/project-report.md

Udacity- deep learning