# Learning to Classify Digit in Real-World Images with Convolutional Neural Network (Report of Deep Learning Assignment)

Han Tang
Student No.: D16129273
Programme Code: DT228A DA
Technological University of Dublin

***Abstract*—A ConvNet model is built from SVHN dataset to classify single digit from real-life images of street number. The model reaches a good performance on the test dataset (an accuracy of 95%). This report is meant to describe the process of preprocessing the image dataset, building the model, evaluate the performance and compare this model with other existed work.**

***Index Terms*—CNN, SVHN, Image Classification**

The Street View House Number(SVHN) dataset is a well-known dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. Lots of Neural network models built from this dataset and reached a relatively good performance. My work is referenced and implemented from multiple works and meant to reach a higher classification accuracy than most of existed works.

This report will describe my work of building and evaluating a convolutional neural network model for this problem in detail. Start from a simple but informative descriptive analysis on the image dataset, then to create divide the dataset to train, validation, and test dataset, then to preprocess the image and modelling. Finally, evaluate the performance of the model on the test dataset and discuss if there can be any further improvement on the model.

## I. Model Design

### A. Dataset Retrieving

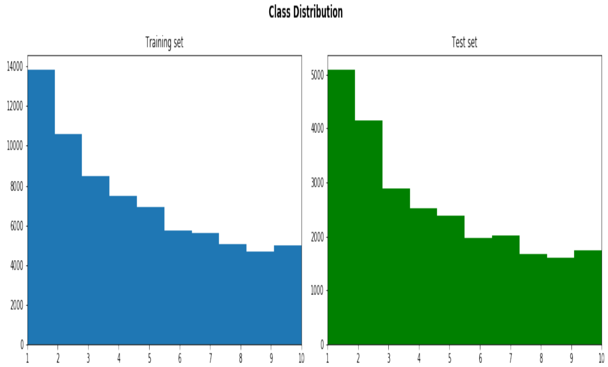
There are two datasets both called SVHN, one is full-image dataset, the model shall detect multiple digits and classify them. The other one is simpler, a 32 by 32 cropped dataset. The model is only supposed to classify the digit in the center. The model described in this report is trained from the 32 by 32 single label dataset. Thus the model will classify a input as only one class.

There are three datasets in the 32 by 32 Cropped digits part. 'train_32*32.mat','test_32*32.mat', and 'extra_32*32.mat'. Only the train and test dataset will be chosen for modelling, as the extra one is too large, makes the modelling process too time consuming without improving the performance significantly.

Fig. 1. An Image Example(class = 0)

### B. Descriptive Analysis

The training dataset contains 73257 samples, and the testing dataset contains 26032 samples.



It is found the class distributions of classes for the two datasets are in similar pattern. The distribution has a positive skew, indicates there is an overweight of smaller values in the dataset.

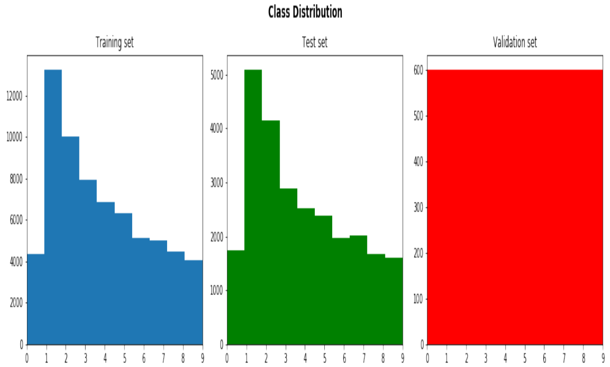
### C. Data Division and Preprocessing

All images of zero are labelled as 10, thus the first step is to relabel them to 0.

Then I randomly selected some data from the training dataset to build a validation dataset. The validation dataset is balanced (all classes are equally presented) for the purpose of diminishing the influence of unbalanced class distribution in the training dataset, according to Chang [1] and Lee [2]

As a result, 6000 samples are randomly selected from the training dataset (600 samples each class) to build a validation dataset.

Those samples chosen into validation dataset are removed from the training dataset.



*1) Transfer to Grayscale:* For the purpose of diminishing the size of dataset. The RGB image data is transferred to grayscale image data. The dimension contains 3 digits (represent the degrees of Red, Green, and Blue) is transferred to 1 digit represent the scale of gray. The size of the dataset is diminished, and the information within the image is preserved in this way. The equation of transformation is below.

$$Y' = 0.2989 * R + 0.5870 * G + 0.1140 * B \qquad (1)$$

*2) One Hot Encoding:* Then One Hot Encoding is applied to the label values.

*3) Nomalization:* Mean subtraction and normalization are applied to the data, according to the report of Goodfellow [3].

### D. *Model Architecture*

After data preprocessing, I designed the architecture of the convolutional neural network.

There are 6 convolutional layers, with a max pooling layer after two convolutional layers. Then the feature selected data are sent to full connected neural networks.

The image is firsly sent to a convolutional layer with 32 filters, the size of the convolutional unit is 3 by 3. The stride is set as default, which is 1. The feature selected images are padded as the same size as the previous. Then the data is normalized and sent to RELU function.

Two convolutional layers with the same design are applied, then the data are sent to a max pooling layer, with a pooling size of 2 by 2.

In this step, 30% of neurons are randomly dropped to prevent overfitting.

Then similar convolutional layers are applied, with the number of filters increase gradually from 32 to 64, then to 128.

The image data are then flattened and sent to a regular densely-connceted Neural Network layer. 30% of neurons are randomly dropped of each training sample.

Then the model sends the data to the output layer, which contains 10 neurons, represent 10 possible classes of the test dataset.

### E. *Model Training*

Cross-entropy is chosen to evaluate the loss, neurons are optimized by Adam Gradient Descent.

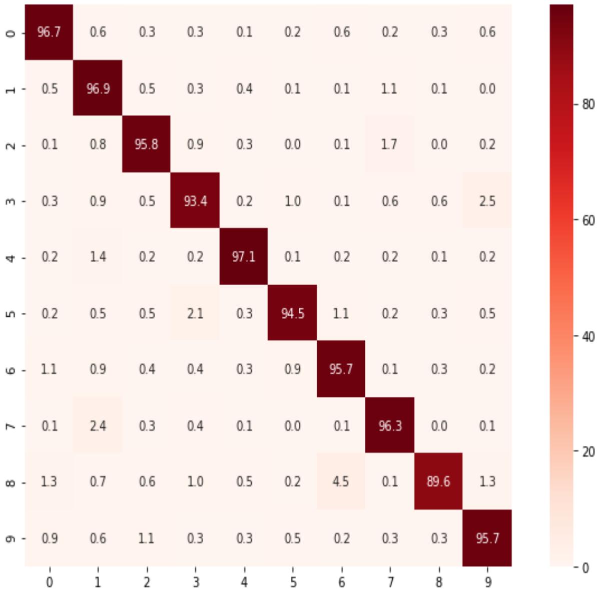
Early stopping method is applied as well, evaluated by validation accuracy of each epoch. In order to prevent overfitting.

## II. Results

The max epoch of training is set to 50, but the training process early stopped at the 16th epoch. Training loss = 0.1516, training accuracy = 0.9561, validation loss = 0.2229, validation accuracy = 0.9412, test loss = 0.2053, test accuracy = 0.9438.

### A. *Confusion Matrix*

The performance of the model in the test dataset is firstly evaluated by a confusion matrix.

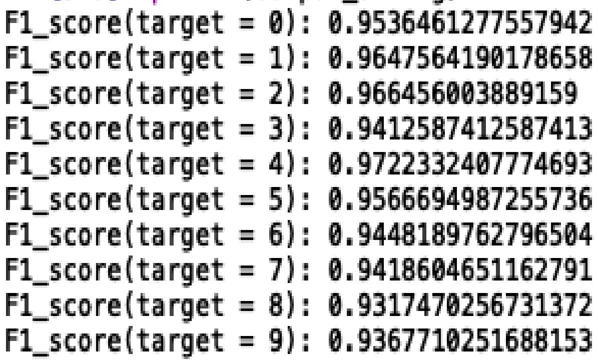


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 96.7 | 0.6 | 0.3 | 0.3 | 0.1 | 0.2 | 0.6 | 0.2 | 0.3 | 0.6 |
| 1 | 0.5 | 96.9 | 0.5 | 0.3 | 0.4 | 0.1 | 0.1 | 1.1 | 0.1 | 0.0 |
| 2 | 0.1 | 0.8 | 95.8 | 0.9 | 0.3 | 0.0 | 0.1 | 1.7 | 0.0 | 0.2 |
| 3 | 0.3 | 0.9 | 0.5 | 93.4 | 0.2 | 1.0 | 0.1 | 0.6 | 0.6 | 2.5 |
| 4 | 0.2 | 1.4 | 0.2 | 0.2 | 97.1 | 0.1 | 0.2 | 0.2 | 0.1 | 0.2 |
| 5 | 0.2 | 0.5 | 0.5 | 2.1 | 0.3 | 94.5 | 1.1 | 0.2 | 0.3 | 0.5 |
| 6 | 1.1 | 0.9 | 0.4 | 0.4 | 0.3 | 0.9 | 95.7 | 0.1 | 0.3 | 0.2 |
| 7 | 0.1 | 2.4 | 0.3 | 0.4 | 0.1 | 0.0 | 0.1 | 96.3 | 0.0 | 0.1 |
| 8 | 1.3 | 0.7 | 0.6 | 1.0 | 0.5 | 0.2 | 4.5 | 0.1 | 89.6 | 1.3 |
| 9 | 0.9 | 0.6 | 1.1 | 0.3 | 0.3 | 0.5 | 0.2 | 0.3 | 0.3 | 95.7 |

It is found the model performs well in predicting most of the digits, reaches a classification accuracy of 0.9438 overall. The accuracy of each label is varied, though most of them are above 93%. The model only fails to give a performance as good as the rest classes on classifying class 8, with a accuracy of only 90%. Nevertheless, this model still performances very good, evaluated by confusion matrix.

### B. *F1 Score*

Then F1 score is also applied to evaluate the model.



```
F1_score(target = 0): 0.9536461277557942
F1_score(target = 1): 0.9647564190178658
F1_score(target = 2): 0.966456003889159
F1_score(target = 3): 0.9412587412587413
F1_score(target = 4): 0.9722332407774693
F1_score(target = 5): 0.9566694987255736
F1_score(target = 6): 0.9448189762796504
F1_score(target = 7): 0.9418604651162791
F1_score(target = 8): 0.9317470256731372
F1_score(target = 9): 0.9367710251688153
```

When evaluated by F1 score, it can be found that the model has very good performances on all the labels, reaches a minimum F1 score of 0.93.

## III. Conclusions

This report describes the process of preprocessing the image dataset, building and training a CNN model, and evaluating the trained model. This model reaches a good performance on the SVHN dataset, as good as manual work. The image can be transferred to grayscale for training, without influencing the performance and reduce the computational cost of training. The architecture of multiple convolutional layers are proved to be effective in helping to reach a high performance.

### A. *Further work*

As mentioned, this problem composed of two datasets. This model is built on the simpler dataset with 32 by 32 images contain only 1 digit. Further work will focus on the full image dataset, aims to build a model can detect all the digits in an image and conduct a multi-nominal classification on the image.

## References

[1] Chang, J. R.(2015) Batch-normalized Maxout Network in Network. CoRR, 1511.02583. url = http://arxiv.org/abs/1511.02583.

[2] Chen-Yu Lee*, Saining Xie*, Patrick Gallagher, Zhengyou Zhang, Zhuowen Tu. In Proceedings of AISTATS 2015 An early and undocumented version presented at Deep Learning and Representation Learning Workshop, NIPS 2014

[3] Ian J. G., Yaroslav B., Julian I., Sacha A., Vinay S. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. url = http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdf.