Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks Review and Implementation

Mert Ketenci, Aijia Gao and Shimeng Feng

Columbia University, SEAS Engineering School

December, 2018

Abstract— **This project aims to reconstruct, implement, analyze, and improve a deep learning (CNN) model to recognize multi-digit number sequence in unconstrained natural photographs [1]. The model is presented by Ian J. Goodfellow et al (2014) [2]. This is a challenging and interesting topic in computer vision community because of the irregular arrangements of the digits, the various factors that affect image quality, and the broad application. In this project, the application is mainly in recognizing the street numbers embedded in street view photos. We first reconstruct the model proposed by the paper and did exploration on hyperparameter tunning and model architecture, trying to advance the detection accuracy and the overall flexibility of the prediction. The reconstructed model was trained and tested on Public Street View House Numbers (SVHN) dataset. Our model has reached a best testing accuracy of 82% with about 3 hours of training time on 33,401 sample images. The final accuracy is lower than the accuracy presented in the paper. This is mainly due to the computation challenges on processing and training on large sets of image samples, which results in a smaller size of training samples used in actual training. With limited computation resource, we also have limited exploration on hyperparameters which may result in a less optimal setting during the training. The final goal is achieved by discovering the most accurate model structure and hyperparameters combination, a satisfactory accuracy considering limited computing power, and additional exploration on models beyond the Google model.**

I. Introduction

Multi-digit numbers in photographs through images captured at street level has a significant usage in modern-day map making and optical character recognition (OCR) on photographs or documenting processing fields. In addition, there are still many challenges from both environmental factors and image acquisition factors affect image quality and prediction results. The paper present by Goodfellow et al.(2014) has discovered an unified approach (Dean et al,2012)[3] that provides a high-accuracy conditional probabilistic model through a deep convolutional neural network.

The goal of this paper is to implement the proposed model on Street View House Numbers Recognition, and validate the given accuracy. Moreover, our team also tested various hyperparameters, model structure, and additional models to seek potential opportunity for improvement.

II. Literature Review and Related Works

Multi-digit recognition is a well known hard problem. There have been many different approaches to that problem. Currently, there are two major methods to approach this problem:

bottom-up and top-down. For bottom-up approaches, many of them consist of detecting the edges in the images and separating individual digits of the full sequence from the image to formulate an input. In such networks, the inputs are images of independent digits. For example, Min Jeon et al. discusses this approach in their paper *Real-Time Multi-Digit Recognition System Using Deep Learning on an Embedded System* (2018) [4]. The methodology is then to formulate a network with an output layer of 10 neurons and build an edge detector for each digit. Yet, this methodology can be interpreted as recognizing a single digit. A similar methodology was applied by Yann LeCunn at Bell Labs to recognize handwritten zip code digits on the paper *Multi-Digit Recognition Using A Space Displacement Neural Network* (1992) [5].

However, the paper proposed by Google approaches this problem in a top-down fashion. The model eliminates the part of separating digits from each other and takes the image of a full sequence as an input regardless of their length. The edge detection now happens inside the convolutional layers and the digits/text that are processed are passed to dense layers. This enables the network to operate faster and uniform but also increases the number of output neurons. A more recent paper by Cheng Zhanzhan et al.[6] have proposed an Arbitrary Orientation Network (AON) to better recognize irregular or not tightly-bounded, nor horizontal or frontal text. This model is able to extract the individual features first using the AON. Then it applies the filter gate (FG) to generate an integrated feature for the full sequence, which is used prediction. This method eliminates the process of localization. In addition, for text detection specifically, sometimes CNN and RNN are combined together to leverage conditional probability to improve accuracy.

There are many other methods exists in this field, but the model presented by Google provides a relatively straightforward architecture and a high prediction accuracy.

III. Summary of the Original Paper

A. Methodology of the Original Paper

Google has proposed a unified approach to recognizing multi-digit numbers. CNN is used first to transform image features into one feature vector. This overall feature vector is then used in 6 classifiers to predict individual digits and sequence length. Specifically for SVHN dataset, the model is designed as below: 11 hidden layers in total, which includes 8 convolutional hidden layers, two densely connected layers, and 5 locally connected layers, one for recognizing each digit within the sequence (assuming a maximum length 5 digits) and a layer for predicting the length of the sequence. All of the convolution layers have zero padding to preserve the shape of the input. All convolution layers have a kernel size of 5*5. The first four CNN layers have [48, 64, 128, 160] as the number of filters, and the rest of the convolution layers have 192 as the number of filters. A max pooling layer of 2*2 is used in each layer. The stride of max-pooling alternates between 2 and 1 at each layer which reduces the size of input in half of the layers. The output from the convolution layers is then fed into two dense layers, which contain 3072 neurons at each layer with drop-out applied.

The output of the fully connected dense layers is then used as the input to 6 Softmax classifiers recognizing each individual digit(S_i) within the full sequence and the length (L) of the overall sequence (S). The prediction is then made by picking the sequence with the highest overall probability. The overall probability is calculated as below:

$$P(S = s|X) = P(L = n)\prod_{i}^{n} P(S_i = s_i) \qquad (1)$$

The model is evaluated against three metrics: sequence transcription accuracy, coverage at certain confidence threshold, and a digit -level accuracy. The transcription accuracy and digit-level accuracy can be calculated by the formula (2). The difference is that the transcription accuracy is defined by the ratio of correctly identified sequence over all sequences. The sequence is correctly identified only when all the digits are correct and the length is correct. The digit-level accuracy is the ratio of correctly recognized individual digit overs all digits appeared in all street numbers. It did not consider whether the length is correct. The third metric is coverage at a certain confidence threshold. To evaluate coverage, the model returns a confidence value about its prediction and prediction that have confidence level lower than a threshold will not be taken. The coverage is then the ratio of accepted prediction overs a total number of images.

$$Accuracy = \frac{\text{No. of correctly identified full sequence}}{\text{Total Number of sequnece}} \tag{2}$$

The overall architecture of the original paper is represented as Fig. 1:

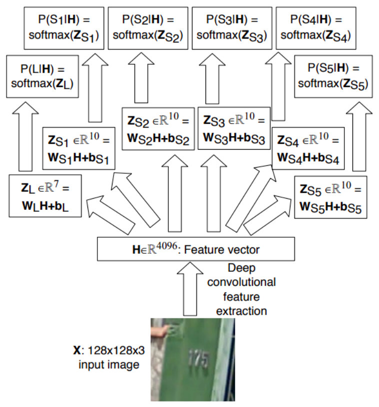


Fig. 1: Model for detection multi-digits number

In addition to the SVHN dataset, the Google team has also tested their model on relative harder Google internal street view data and CAPTCHA puzzles data. These two datasets have introduced features such as text recognition and certain levels of rotation. Within each data, modifications and changes have been applied to the model structure to adjust to the characteristics of each dataset.

However, our team has discovered that there are several limitations of this model:

- The paper assumes that there is no overlapping between digits for the SVHN data while implementing the model
- The paper mentions that there are factors other than the depth of the network that would improve the performance on this task, but there was no further exploration on those potential factors
- The paper only performed the basic image processing by subtracting the mean of each image and basic data augmentation. However, no additional efforts on feature engineering have been applied, such as PCA
- There is no specific running time for the Google Internal data and the CAPTCHA data, so it is hard for readers to understand their efficiency across datasets
- The paper does not mention the effect of different optimizers on model performance
- The calculation details of the prediction accuracy is not well explained, which makes it hard to for readers to replicate the model

B. *Key Results of the Original Paper*

Through the model, three datasets are tested and each of them has obtained the accuracy as the following:

- The Public Street View House Numbers: the transcription accuracy is 96.03%, the coverage at 98 % confidence threshold is 95.64%, and the character-level accuracy is 97.84%.
- The Google internal data, the transcription

accuracy is 91%, the coverage at 99 % confidence threshold is 83%, or 89% at 98% threshold.
- CAPTCHA puzzle dataset: 99.8% transcription accuracy

Overall, the Google team discovered that deeper CNN network provides better performance. The early layers perform localization and segmentation tasks. Also, there is a limitation within the model, such that it is efficient for shorter sequence recognition but for longer sequence, the model needs to have more locally connected layer which may be restricted by computation resource.

IV. Our Methodology

This session describes our approach to implement Goodfellow et al.'s model, the detailed structure, and various technical difficulties our team has encountered during training and implementation.

A. Design and Implementation

Overall Work Flow: Following the original paper, our overall work flow is described as below:

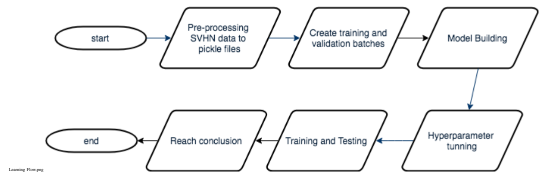


Fig. 2: Overall Workflow of the project

Data Processing:

The data has been downloaded from the SVHN website [7] and our team used training and test data under format 1. The images are marked with blue boxes indicating the location of each individual digit within the image. The data is processed in the following way: load the data and then read the digitStruct.mat file which provides the coordinates of the blue rectangular bounding box. Then the coordinates of each bounding box are used to construct a larger box that encloses all the individual boxes (labeled as a red box as shown in Fig. 3). Then the red box is expanded by 30 % on each edge to include a relatively larger area to ensure all the necessary information will be preserved during the cropping process. The newly-composed rectangle is used to crop the original image and then the cropped image is re-sized into a 54*54*3 format. The labels have been converted from 1-10 tags into 0-9 tags for each digit and then combined the individual label into a string for each picture. For example, label '1' and label '9' becomes label '19' for an image. A pickle zip file is generated for training dataset for further usage. Then the similar process above has been repeated on the testing dataset.

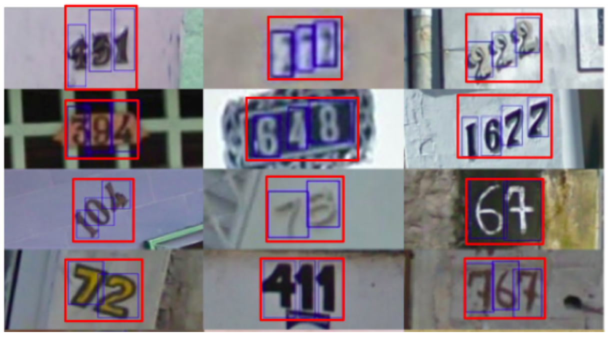
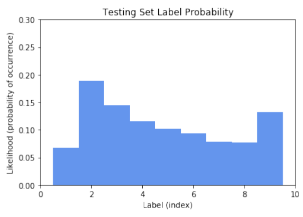


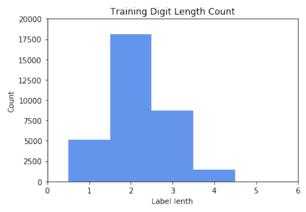
Fig. 3: Bounding Box for Image Cropping

After we have successfully generated the data file, our team have also deep dived into the nature of the data to gain a better understanding of the population. Here are some of our key findings:

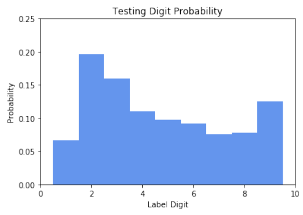
The distribution of length and the digits are shown in Figure 4:



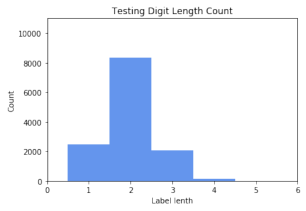
(a) Digits distribution of training data



(b) Length distribution of training data



(c) Digits distribution of testing data



(d) Length distribution of testing data

Fig. 4: Data Length and Digits Distribution

From the above distribution we can conclude that within the training data, the most frequently appeared street number length is 2. There are some outliers, such as there is one image has a street number of length 6 and 9 images have a length of 5. Digit 2 and 9 appear most frequently within the training data. For testing data, we observe a similar trend as training, except there is no street number with a length of 6. The original paper has also indicated a similar discovery as their assumption of having the longest the sequence n as 5. However, in order to cover a broader possibility, our team has decided to include the sequence n at most 6 based on the dataset. This provides additional flexibility to our model.

Implementation:

Our model has 8 convolutional layers with the same dimension as indicated in Goodfellow's model. Within each convolutional network, batch normalization is applied. After the CNN layer, two fully connected layers with units 3072 implemented with drop out. Then 6 locally connected layers are created to perform classification for digits 1 to 5 of the full sequence and the length of the sequence. A total of 6 softmax classifiers have been applied and a total loss is calculated by adding the loss of all 6 softmax classifier. Then Adam optimizer is applied to minimize the overall loss through iterations.

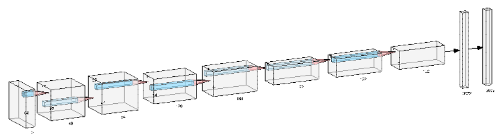


Fig. 5: Model Structure

Prediction is determined by calculating the total probability of the full sequence using the following formula (Table 1). For each digit of the sequence (digit 1 to digit 5), we first choose the digit to be the number that has the highest probability $P(S_i)$. Then we sum up the log of $P(L)$ and log of $P(S_i)$ for L = 0 to 5 and more than 5, which gives the probability of the sequence with different length, Then we predict the sequence to be the one with the largest probability. An evaluation function is implemented to compare predicted sequence with actual street number and calculate the transcription accuracy defined in Google's paper.

TABLE I: Prediction Calculation

L	Log P(S) Prediction
0	$logP(L=0)$
1	$logP(L=1)+logP(S_1)$
2	$logP(L=2)+logP(S_1)$ $+logP(S_2)$
3	$logP(L=3)+logP(S_1)$ $+logP(S_2)+logP(S_3)$
4	$logP(L=4)+logP(S_1)+logP(S_2)$ $+logP(S_3)+logP(S_4)$
5	$logP(L=5)+logP(S_1)+logP(S_2)$ $+logP(S_3)+logP(S_4)+logP(S_5)$
>5	$logP(L=6)+logP(S_1)+logP(S_2)$ $+logP(S_3)+logP(S_4)+logP(S_5)$

The total log probability is calculated by adding both the log of the probability of different length function and log of the probability of corresponding digits

B. Challenges and Problem Faces

Data Processing:

The input data was composed of street numbers of different sizes. The size of the training data is more than 33,000 images. During the data exploring step, our team has tried two different methods. The first approach was cropping each separate digit with respect to their position and 'glue' the image of each digit back as a full image of the street number. Because the location of digits in the image is not always in the same alignment, this methodology could result in the loss of relative position of each digit. Thus, instead of concatenating digits, we end up with the method described in the Data processing section above, which is similar to the method used in the paper. This way the variance in the digit location in original image is preserved.

A deficiency of this methodology is that it will provide inconsistent image sizes as the bounding box size were unique for each image. A uniform size data is required for training, so we have to re-size the image to 54x54x3. Yet, the image quality will be affected after resizing an image that is smaller than 54x54x3. Even some of the original images already have poor quality and are hard to detect by human eyes. Examples could be found as below:

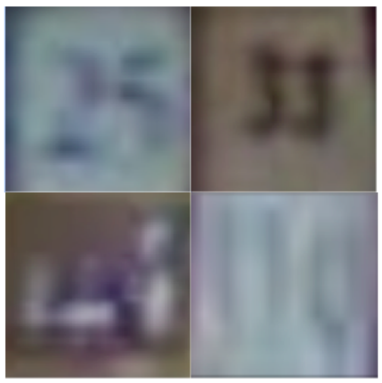


Fig. 6: Example of low quality image after resizing

Model Building and Training:

Another technical challenge our group encountered was determining the dimensions for the neurons of each digit. On the original paper, the dimension was selected as 10 to represent each digit. But there was a possibility of having no digit at some location. Using a street number of length 3 as an example, since the model has 5 classifiers for each digit, the last two are going to force themselves to choose a digit for the 4th and 5th digit as well. In practice, this can make it harder for a model to converge because it forces model to identify non-existed digits. Thus, makes it harder for a model to reduce the loss. Therefore, we decided to change the neuron size to 11 instead of 10 to represent digits between 0-9 and not exits. Therefore, when creating the one-hot encoded labels for digits, we assign 1 to the 11th dimensions indicating that no digit exits in that position.

While creating the training session, we referred to the training functions in our previous homework. One thing we realized was the batches were feed into optimizer following the same order across each epoch. Because of this, our loss always jumps to high values at the start of each epoch and the convergence of loss is slower. To fix this problem, we decided to draw batch randomly from the samples for each epoch. Using this practice, our gradient descends process becomes more stable.

During the initial modeling building phase, the team has to perform multiple training and test to ensure our model structure is correct and identify proper field for improvement. Each iteration takes about 3 hours and the overall task is quite time consuming.

Loss Function:

In addition, interpreting how the loss is calculated in the original model has been a challenge. The main reason for this is the dense layer, which gets inputs from convolutional layers, forks to 6 locally-connected layers. Thus, it is necessary to define 6 different losses and combine them as the total loss to be used in optimization. Figuring this out has been a challenge for our group as we have never dealt with networks that have local loss functions. To be able to come up with a solution to this problem, the following computational graphs is implemented.

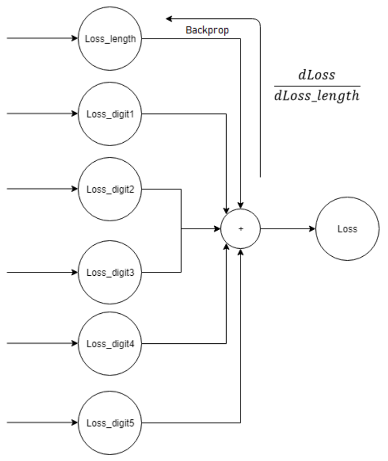


Fig. 7: Computational graph of loss function

Hyper-parameter Tuning:

Hyper-parameter tuning has been a challenging topic since the size of the data is large and it takes long time to even try a few different hyperparemters. To deal with this issue, we selected a portion of the data to optimize our accuracy through tuning the hyper-parameters parameters.

Another challenge of hyperparameter tuning is that effect of one hyperparameter on performance might correlate with other hyper parameters. Thus, it is difficult to select best hyperparameters without considering other parameters. Therefore, we decided to do grid search of the combination of hyperparameters.

Software and hardware difficulties:

Within the original paper, the model has been trained for six days using 10 replicas in DistBelief on the SVHN dataset. This is a huge training effort and it is hard to reach with our current Google Cloud Platform (GCP) instance setup. The infrastructure that we have used in the GCP instance is 1 NVIDIA Tesla k80 GPU, 2 CPUs, and 7.5GB memories. The training processes have been run in parallel between 3 team members. With the above setup, it took about 3 hours to run 100 epochs on 33K images. In addition, the team has encountered technical difficulties with GCP while running the model.

Due to the limitation of the computing power, we have encountered resource exhausted error for certain tasks. For example, with our instance setup, the team was not able to test on the whole 13068 images all at once and obtain the testing accuracy. Therefore, the testing set has been divided into mini-batches and tested individually and average on the accuracy is obtained.

V. Results

A. Overview

In this section, we present the results from the grid search of optimal hyperparameters and the prediction performance of the model trained using the best hyperparameters.

B. Hyperparameters Tuning

In order to find the optimal hyperparameters, we selected three candidate values for each hyperparameters: (0.2, 0.5, 0.8) for dropout rate, (16, 32, 64) for batch size, and $(10^{-4}, 10^{-3}, 10^{-2})$ for learning rate. We then tested out all possible combinations of these three parameters (a total of 27 combinations) on 3000 samples with 100 epochs. The validation accuracy for these sets is summarized in Table 2, 3 and 4. Figure 8 shows the change of accuracy over different learning rate with dropout rate fixed at 0.5. Based on results in the table and Figure 8, we can see that the learning rate of 10^{-4} has consistently the best performance and 10^{-2} learning rate always give the worse results.

As for dropout rate, its effect is more correlated with batch size and learning rate. Figure 9 shows the change of accuracy across different dropout rate at a learning rate of 10^{-4} and Figure 10 has a learning rate at 10^{-3}. It seems 0.8 dropout rate has shown consistently good performance while the accuracy with 0.2 and 0.5 dropout rate fluctuate.

Finally, for batch size, it shows that with a good choice of learning rate, the effect of batch size is not significant. The average accuracy at 10^{-4} for each batch size choice is 56% for 16 batch size, 53.1% for 32 batch size and 52.3% for 64 batch size. Batch size mainly influences the training time. The smaller the batch size, the longer the training time for the same amount of data.

For the epoch size, we observed that the loss and accuracy approach plateau after certain epochs (around 30 when training on the full dataset) and the improvement on the accuracy become very small after that. However, it takes around 3 hours to train for 100 epochs but the improvement over accuracy is minimal. Therefore, we did not try to further improve the accuracy with more than 100 epochs.

Based on the results of 27 tests of hyperparameters, the best setting is 16 as batch size, 10^{-4} as learning rate and 0.8 as dropout rate.

TABLE II: Comparison of validation accuracy using different combination of drop out rate and learning rate using batch size 16

	Drop-out Rate		
Learning Rate	0.2	0.5	0.8
10^{-4}	56.2	54.3	57.5
10^{-3}	45.3	1.7	43.5
10^{-2}	1.7	1.7	1.7

Training time is around 15 mins using batch size 16 for 3000 data.

TABLE III: Comparison of validation accuracy using different combination of drop out rate and learning rate using batch size 32

	Drop-out Rate		
Learning Rate	0.2	0.5	0.8
10^{-4}	51.7	54.0	53.7
10^{-3}	1.7	1.7	45.8
10^{-2}	1.7	1.7	1.7

Training time is around 10 mins using batch size 32 for 3000 data.

TABLE IV: Comparison of validation accuracy using different combination of drop out rate and learning rate using batch size 64

	Drop-out Rate		
Learning Rate	0.2	0.5	0.8
10^{-4}	51.0	53.0	52.8
10^{-3}	45.3	45.5	43.3
10^{-2}	1.7	1.7	1.7

Training time is around 7 mins using batch size 64 for 3000 data.

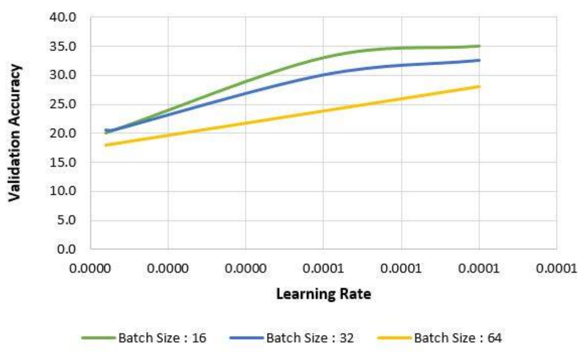


Fig. 8: Dropout Rate $= 0.5$

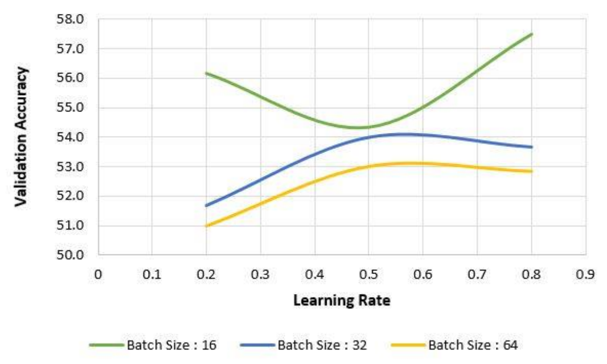


Fig. 9: learning Rate $= 10^{-4}$

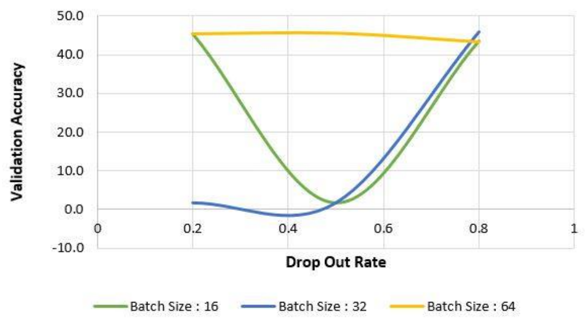


Fig. 10: learning Rate $= 10^{-3}$

C. *The Best Model Results*

Following the model architecture proposed by Ian J. Goodfellow et al., we have implemented the model and trained the network using the hyperparameters selected from the grid search. With 16 as batch size, 10^{-4} as learning rate and 0.8 as dropout rate, we were able to reach 83% validation accuracy and 82% test accuracy with 100 epochs. The total training time is around 3 hours and 15 minutes. However, we observed that after around 50 epochs, the loss reduced to nearly zero. Fig. 11 shows the loss over iterations and Figure 12 shows the accuracy over iterations. Since the cross-entropy loss function cannot have negative value, the gradient descent optimizer will not be able to make much improvement on the model when a loss is nearly zero.

Loss	Test Accuracy	Validation Accuracy
4.5×10^{-7}	82.0%	82.7%

TABLE V: Performance of our best model

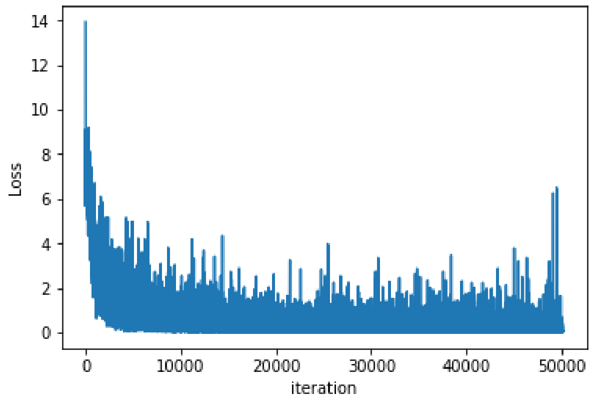


Fig. 11: Model Loss Over Iteration During Training

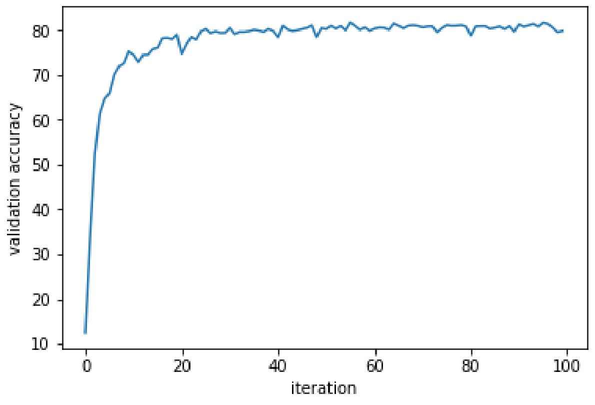


Fig. 12: Model Accuracy Over Iteration During Training

D. *Comparison of Results*

The paper reports a 96% accuracy over publicly available SVHN dataset. Similar to the paper, we only considered an image to be correctly classified when the model correctly predicts the full sequence of the street number. Street numbers that are classified partially correctly, meaning some of the digits of the street number are classified correctly, are still considered an error.

Compared with the performance reported by the paper, we did not obtain the same performance. These may due to several reasons. First, during training, Google runs 10 replicas of the model and spent 6 days on training. With better computational power, they were able to explore more choices of hyperparameters. Therefore, the hyperparameters they used during training may be better than the sets we explored. Second, although the model improvement becomes slower after several iterations, it is still improving. However, due to the time limit, we were not able to train the model for a long enough time. Finally, Google trained their model using about 200k street number images from the public street view house numbers datasets, while we only used around 30k training samples. This is probably the most important factor that influences the model performance.

Consider the limitation of training samples and computational power, we believe an accuracy of 83 % is satisfactory.

E. Discussion of Insights Gained and Future work

- Insights on training sample

  To deal with the constrain on sample size, we could use data augmentation to create artificial samples. Yet, in the paper it has been indicated that data augmentation only increases the accuracy by 0.5%. Thus, we did not apply data augmentation. The source website of the SVHN dataset also provides an additional package containing around 200k images. Due to the limitation of computing power, we were not able to extract all those images and process them. Even we extract the pixel values of the images, the model cannot be trained due to the resource limitation of GCP.

  To cope with the issue of limited computation resources, another alternative to reduce the computation time is by PCA. PCA analysis can be applied to reduce the dimension of data by keeping only features that have high variance. Thus, even though the dimension of sample size is increased to 200k, the dimension of the feature will be decreased by PCA without much loss of information of the original images. The model presented by Google was trained on all available images (including the original 30k training images and the additional 200k images), so we believe our model performance can be improved further with the additional 200k images.

  In addition to the above insights, our team has also cast some concern on the distribution and variety of samples. As mentioned above in the Data Processing section, the samples fed into the model have more two-digit street numbers and digit 2 appears more frequently than other digits. Thus, due to the imbalance of class distribution, it is possible that the model is not truly ”learning” the features in images to predict the street number, but rather simply predicting the digit or length with the highest weight in overall population. Further testing may be valuable by using samples that are drawn evenly from different classes to conclude if the biased data has additional effect on the prediction accuracy.

- Over-fitting in neural network

  During the process of building the model, our team first test the model on the training data to see if our model can at least learn the training images and predict well on them. This provided a

98% accuracy, which means the model fit the training data closely. However, once we have split the training data into 80% training and 20% validation, the accuracy dropped to around 74% without optimizing the hyperparameters. This gave us an insight that the model maybe over-fitting during the training process.

One thing the team has tried is to increase the drop-out rate. As mentioned in the Hyperparameter Tuning section, three drop-out rates have been tried (0.2, 0.5, 0.8) on a set of 3000 data. We have discovered that the 0.8 provide consistent good prediction accuracy. Using high drop out rate means that during training, majority of the neurons will be dropped out which further control the over-fitting of the model. The fact that a higher drop-out rate provides a more accurate result further confirmed our hypothesis that our model is over-fitted and more regularization should be applied through the process. In the paper, it is also mentioned that Google has observed overfitting when the model is trained using SVHN dataset.

- Randomization of training batch

  As we have discussed above, during initial training, our team encountered cases showing high loss and slow reduction in loss during gradient descent steps. We have hypothesized that this is because the training batch was feed to model in the same order throughout each iteration. This may cause the loss to overshoot on every new epoch. Thus, we change the data feeding mechanism such that at each epoch, batch samples are drawn randomly from the whole training dataset. We saw the loss reduction during gradient descent becomes more stable and we were able to reduce the loss to a smaller value during training.

- Hyper-parameter optimization

  As mention in the Result section, the effect of drop out rate and batch size are still not fully uncovered. We did not have a conclusion on why model with 0.2 and 0.5 drop out rate reveals opposite behavior for different batch sizes. This correlation between batch size and drop out rate can be further explored in future by performing testing using more choices of drop out rate and batch sizes.

  Our team is also interested in discovering more combination of hyperparameter by leveraging the results of *Algorithms for Hyper-Parameter Optimization* (2011) [8] in the near future. As mentioned in the paper, the random search and two new greedy sequential methods could efficiently support the search of the optimized hyper-parameters.

- Exploration on alternative models

  First, our team modified the original model to seek a higher accuracy. The paper mentioned that while training on the Google internal data, the fully connected layers with 4096 units per layer behave the best comparing to other models. Our team has hypothesized that this is because, with more neurons, the network might be able to recognize more complicated features or overcome the low quality of images. Thus, we have updated the units from 3072 units into 4096 units. However, using this updated model to train on the 33K data did not provide any significant improvement in accuracy. The reason behind this could be the generalization ability of our model is limited by our training sample size.

We also tried to approach the problem using alternative models. One of the approaches was trying to formulate the problem as a regression problem. Thus, instead of have 6 outputs, the network will predict a numeric value for the street number and minimize the mean squared loss between the predicted value and actual street number. Then, the final prediction is made by the rounding the predicted number to the nearest integer. The srtucture of this model is shownin Figure 13.

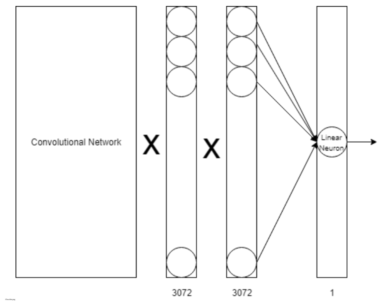


Fig. 13: Regression Classifier

From this model, we observed that using regression to approach this problem is not very feasible due to the stricter requirement on the loss value. For example, in order to correctly predict 1000 as street digit number, the output from the network have to be between 999.5 and 1000.5, which means a very small loss value is required. It is generally hard for optimizer to reach this small value for each prediction.
Another model we tried is that instead of having 6 classifiers, we use one classifier but have 100 class to choose. For a street number of 2 digits, there are 100 classes. Each number is a class. The two dense layers will then be connected to softmax layer with 100 neurons. This model can be used to predict street number with a maximum length of 2 digits. We showed that this network can classify 2 digit numbers well.

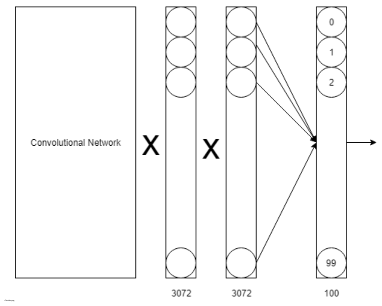


Fig. 14: Classifier for two digit street number

However, this model has a big limitation in application due to exponentially increase in output layer size for the longer street number. Thus, it gets exponentially harder to train this network. However, we were able to get good accuracy on classifying two digit samples during our testing. The results we obtained showed that this model had the potential do short sequence recognition. However, due to the large size of output neurons, this model could also suffer from overfitting.

- Insights on running time

  Using larger batch size during gradient descent can speed up the training process. However, this may impose an issue if memory resource is limited. We have tried a larger batch size like 100 and 200 on GCP, but GPU memory has exhausted due to the large size of tensors. Furthermore, we observed that if the drop-out rate and learning rate is selected properly, the difference in accuracy is not significant using different batch sizes. Therefore, if computation time is an important factor, one may want to sacrifice some performance by choosing a larger batch size, while balancing between running time and memory usage.

VI. Conclusion

In conclusion, we reproduced the model from the paper *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks* by Ian J. Goodfellow et al. (2014) and obtained 82% testing accuracy under the limitation of computation resource and sample size. In addition to successfully reconstructing the model, we have gained insights on the effects of hyperparameters, randomization of training batches, and over-fitting issue commonly present in a neural network. Beyond the model presented by Google, we also sought our own ways to formulate solutions for the multi-digit recognition. We tried to formulate the classification problem as a regression problem and observed that it imposes a strict requirement on the loss value which is hard to achieve. Another approach we used is to treat each street number as a class. We tried our model on recognizing the two-digit street number (which have 100 class) and observed that the results of this model were promising. Nevertheless, with an increasing number of digits. the size of the output neurons of the model increased exponentially so it may not be feasible for long street numbers. Future work has been considered for discovering the effects of hyper-parameters and applying PCA for large sample set training.

VII. APPENDIX

A. Team member contribution

Overall, the work has been distributed evenly across each member. Mert and Aijia collaborated on the model building, hyper parameter tuning and training. Shimeng has worked on data pre-processing and loading part. The team also brainstormed together for other possible methods on street number classification and Mert implemented our ideas and did preliminary testing. The report is also a collaborative work of all team members.

Acknowledgment

We would like to thank Professor Zoran Kostic and all the teaching assistant's support throughout the semester and this specific project. In addition, we would like to express our appreciation towards Google Cloud Platform for the credit that we have applied through our training process. At last, we would like to thank Columbia SEAS Engineering School for the opportunity of the Neural Network and Deep Learning course.

References

[1] https://bitbucket.org/ecbm4040/2018_assignment2_sf2911/src

[2] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In Proc. ICLR, 2014.

[3] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large scale distributed deep networks. In NIPS'2012.

[4] Jeon, Hyung-Min et al. "Real-Time Multi-Digit Recognition System Using Deep Learning on an Embedded System." IMCOM (2018)

[5] Ofer Matan, Christopher J. C. Burges, Yann Le Cun, and John S. Denker. 1991. Multi-digit recognition using a space displacement neural network. In Proceedings of the 4th International Conference on Neural Information Processing Systems (NIPS'91), J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 488-495.

[6] Cheng, Zhanzhan et al. "AON: Towards Arbitrarily-Oriented Text Recognition." CVPR (2018)

[7] data is gather from this website: http://ufldl.stanford.edu/housenumbers/

[8] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11), J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.). Curran Associates Inc., USA, 2546-2554.