Multi-Digit Number Recognition from Street View Imagery Using Deep Convolutional Neural Networks

E4040.2018Fall.RECO.report

Phoenix Ma jm4743, Yihang Ding yd2459, Yiqi Sun ys3127

Columbia University

Abstract

This project aims at recognizing multi-digit from street view which is challengeable due to high variability in wide. We introduce a new preprocessing method by applying Fourier transform and Haar wavelet transform to the image. The evaluation accuracy is improved from 94.60% to 95.14%.

1. INTRODUCTION

Recognizing multi-digit numbers in photograph at street level[1] is an important component to improve nowadays map. In this paper report, we reproduce the paper provided by Google Street View and reCAPTCHA Team and improve the result by adding wavelet transform on the image, in order to recognize multi-digit numbers from Street View panoramas.

The main challenge of the project is that in an outdoor environment, the visual appearance of the text will be influenced by environmental factors such as lighting, shadows as well as by image acquisition factors such as resolution, blurs. Also, different colors of digits and backgrounds, along with different fronts, orientations, styles and character arrangements bring challenges to the recognition problem.

In this project, we tried to do the following two parts of work:

1. *Reproduce the model* provided by original paper which use deep convolutional neural network that operates directly on the image pixels with a model configured with multiple hidden layers.
2. *Introduce a new preprocessing method* by introducing Fourier transform and Haar wavelet transform on the image and use both spatial and spectral information to train the model. By adopting this method, the limitation of fixed convolutional kernel size could be avoided and the evaluation accuracy is improved from 94.60% to 95.14% by using exactly the same model provided by the original paper.

2. SUMMARY OF THE ORIGINAL PAPER

2.1 Methodology of the Original Paper

The original paper learns a model of $P(S|X)$ by maximizing $log[P(S|X)]$ on the training set. The architecture consists of eight convolutional hidden layers, one locally connected hidden layer and two densely connected hidden layers.

The size of convolutional kernels are 5×5, with max pooling window size is 5×5 for each convolutional layer. The numbers of units at first four layers are [48, 64, 128, 160] and 192 for other layers. Also, fully connected layers contain 3072 units each.

The block diagram is shown in *Figure 1*. The inside function of convolutional layer and fully-connected layer are shown in *Figure 2* and *Figure 3*.

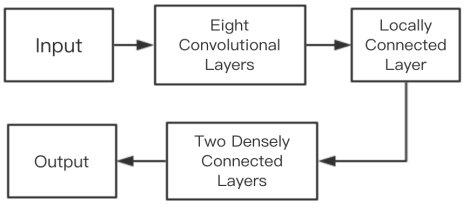


Figure 1. Model Architecture

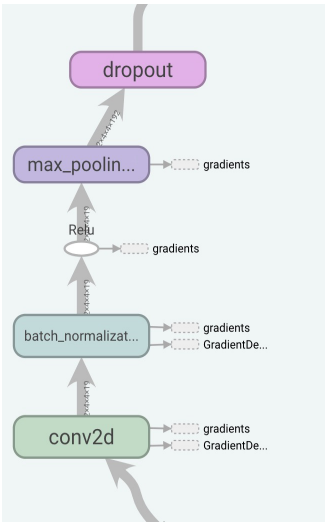


Figure 2. Convolutional Layer Flow Chart

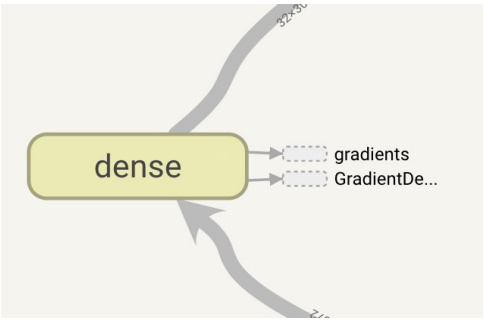


Figure 3. Dense Layer Flow Chart

2.2 Key Results of the Original Paper

The original paper is able to achieve an evaluation accuracy of 96% on multi-digit recognizing problem which doesn't take six-digit number into consideration.

3. METHODOLOGY

The original paper adopts a Convolutional Neural Network to realize the multi-digit recognition from street view. The paper built a 11-layer depth network (8 convolutional layers and 3 affine layers) in total and asserts that prediction result will be better with deeper network applied.

As the original method is mainly built in image convolutions and the depth of the original network is not large, we firstly rebuilt the network and then adopted new analysis/modification in model design.

Intuitively, we built a much deeper network. Secondly, we applied the convolution with taking the spectral domain analysis into consideration. Besides, we also modified the network by adjusting the parameter slightly. Finally, we compared totally the three different applications with the original work so that we can have a much deeper understanding about multi-digit recognition.

3.2 Methodology of Our Image Preprocessing

In traditional CNN computation, we find that the (1) kernel only works on each pixel, namely, the kernel applied in convolution try to find the feature in spatial domain without taking spectral information into consideration. Besides, it cannot be denied that (2) the hyperparameter of convolution kernel size is determined before the experiment and that (3) much information redundancy exists in RGB image format. Thus, we tried to preprocess the image to get the spectral information by 2-D spectrum calculation and create more image information through Haar wavelet transform.

Figure 4. Original RGB Image

To comply with the size of the convolution kernel in previous model, we still set the images with 3 channels as inputs. The first channel with the shape of 64×64, is used to save the gray image of the original image by calculating the weighted average of the original RGB image using eq (1).

$$Gray = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B \tag{1}$$

Figure 5. The grayscale version of the RGB image

After acquiring the first layer, we use grayscale image to calculate the Fourier transform by using eq.(2). and set the output as the second layer.

$$F(k,l) = \sum_{m=0}^{63} \sum_{n=0}^{63} f(m,n) e^{-j2\pi\left(\frac{km}{64} + \frac{\ln}{64}\right)} \tag{2}$$

This application is motivated by some interesting works done by [2], which mainly apply the CNN to audio signal processing by setting the image of spectrum or spectrogram as the input to learn to do the prediction. Thus, we also want to apply the spectrum analysis to the image and then provide more information of spectral domain.

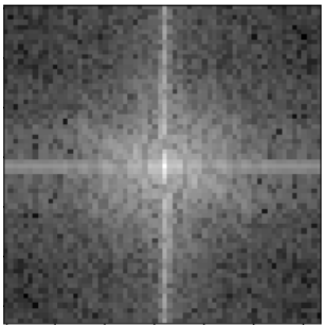


Figure 6. The spectrum of the grayscale image

Figure 7. Image after Haar wavelet transform

Finally, we applied Haar wavelet transformation to the images in order to build our third layer. Take an 8×8 image as an example, wavelet transformation of the image can be described as following where I_G and W_4 represents the grayscale image of layer 1 and the data transformation matrix respectively.

$$Wavelet = W_4 I_G W_4^T \tag{3}$$

$$W_4 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \tag{4}$$

Consequently, according to the transformation matrices' structure, the transformation in matrix form can be described as

$$\begin{aligned} Wavelet = WI_G W^T &= \begin{bmatrix} H \\ G \end{bmatrix} I_G \begin{bmatrix} H \\ G \end{bmatrix}^T \\ &= \begin{bmatrix} H \\ G \end{bmatrix} I_G \begin{bmatrix} H^T \\ G^T \end{bmatrix} \\ &= \begin{bmatrix} HI_G H^T & HI_G G^T \\ GI_G H^T & GI_G G^T \end{bmatrix} \end{aligned} \tag{5}$$

In this equation, $HI_G H^T$ is an approximation of the entire image. $HI_G G^T$ holds information about vertical in the image, where a large value indicates a large vertical change as we move across the image and small value indicates little vertical change. Also, $GI_G H^T$ holds information about horizontal changes in the image, as a counterpart of $HI_G G^T$. Finally, the $GI_G G^T$ represents the diagonal difference of the image.

The application of wavelet transform is motivated by the algorithm of wavelet transform and is designed to solve the limitation caused by the preset kernel size of the convolutional layers. Generally speaking, wavelet transform works as a window where the frequency band can be adjusted according to the changes in the spatial domain and the final output represents the information in both the spatial and spectral domain.

On the other hand, the convolution kernel is designed to extracted the feature from every selected area but it does not work in every row or column or adjacent pixel. If we want to apply the kernel to such condition, the size of the kernel needs to be adjusted from image to image and most parts in the convolution kernel will be zero, which is a great resource waste in neural network.

Thus, we want to apply the data set with three method mentioned above to avoid the redundancy of the image and extract more information from the spatial domain and the spectral domain.

3.1. Objectives and Technical Challenges

The objective is to recognize multi-digit numbers from street view imagery and reach a high degree of accuracy, in which situation only if the whole sequence of numbers is predicted correctly, we value the result as correct. However, in the real world, the quality of images is not ideal all the time.

The challenges arise from the quality and variety of street view imagery. Firstly, the model need to be aware of the length of number sequence. In some situations, one or more numbers are not as clear as other numbers due to different filming angles. Secondly the representation of numbers varies under different weather and time. Due to the lighting, shadow specularities and occlusions, same numbers may result in different images. Thirdly, the images are also influenced by image acquisition factors such as resolution, motion and focus blurs.

3.2. Problem Formulation and Design

The problem is represented as a probabilistic model of sequences of numbers given by images. S represent the output sequence and X represent the input image. The objective is to maximum $log[P(S = s|X)]$ in order to learn the model of $P(S = s|X)$.

S is a sequence of length L and $S_1 \dots \dots S_L$ are elements of the sequence. In our problem, the identification of every single digit is independent of each other. Thus, we use the product of probability of each digit, along with the probability that the sequence length is L, shown as follw:

$$P(S = s|X) = P(L = n|X) \prod_{i=1}^{n} P(S_i = s_i|X) \quad (6)$$

$$log[P(S = s|X)] = log[P(L = n|X)] + \sum_{i=1}^{n} log[P(S_i = s_i|X)] \quad (7)$$

At test section, we predict

$$s = (l, s_i, \dots \dots, s_l) = argmax_{L,S_1,\dots\dots,S_L} log\, P(S|X) \quad (8)$$

Because the result of original paper is already satisfying, we decide to use exactly the same architecture. However, the original paper tried to use the raw images and just reshaped the image. Now we are introducing an additional image preprocess method before training, which results in a better accuracy.

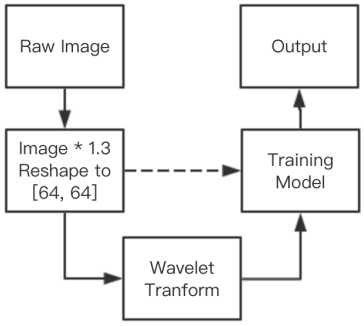


Figure 8. Architecture

Our software contains 4 parts: 1. dataset download and extraction; 2. dataset preprocessing and convert to TFrecord file; 3. CNN model implementation; 4. CNN training and evaluation.

4. Implementation

In this section, we will first explain the architecture of our network in section 4.1 and then shows the algorithm network design detail.

4.1. Deep Learning Network

1. Architectural block diagram(s).

   Our architecture consists of a wavelet transformation preprocessing unit, 8 convolution layers, 1 locally connected hidden layer and 2 fully connected dense hidden layers.

   Each convolution layer consists of a convolution layer, a max pooling layer, a batch normalization layer and a dropout layer. All convolution layers use ReLU as activation function. In our best model, the number of the feature map in each convolution is [48, 64,128,160,192,192,192,192], with the convolution kernel size of 5. Zero-padding is used in all convolution layers to preserve size. All pooling layer have the step of 2 and the stride shifts between 2 and 1 for each layer to avoid overly reduce the size of representation. The fully connected layer has 3072 units each, the activation function is also ReLU.

2. Training algorithm details

   The basic idea of our training algorithm is to set a fixed number of epoch, at each epoch the model is trained by SGD algorithm with the batch size of 32, after the entire epoch is processed, we use the entire test set to validate, and save the model that has the highest validation accuracy. To prevent resource from exhausting, we set the batch size in validation to be 128, but the data is sequentially sampled. The reason for doing validation after the entire epoch is to save time, as the test set has more than 13,000 raw images. In our case, the number of epoch is set to be 20.

3. Data used

   We used the Street View House Numbers (SVHN) dataset[3], which is a public dataset containing more than 240k colored house numbers images with variable-resolutions.

   We preprocess the dataset in the following way:

   Firstly, find the small rectangular bounding box which contains individual character bounding boxes and expand both x and y of the image to 1.3 times of the original value.

Secondly, chop the image into 64*64 and every time data is needed, produce a random 54*54 sub-image out of 64*64 image.

Thirdly, use wavelet transform to preprocess the input image.

4.2. Software Design

Model and Flowchart

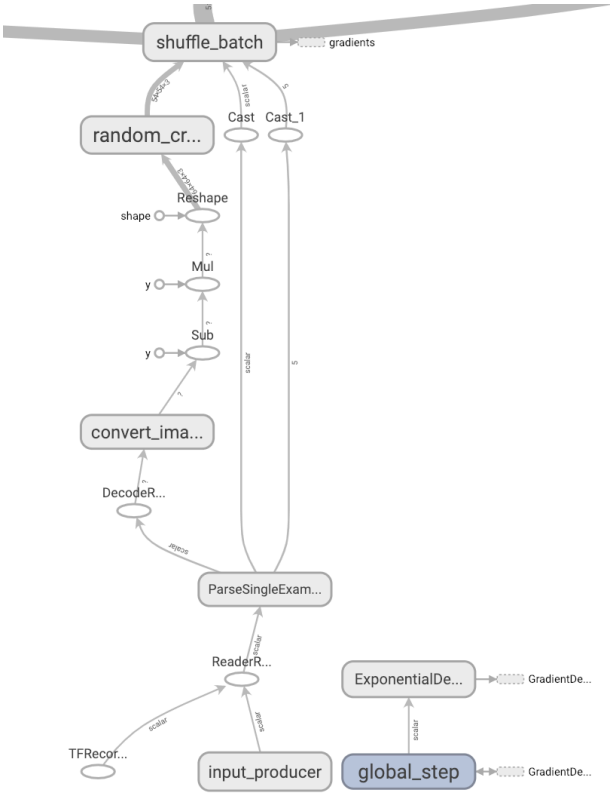


Figure 9. Input Generation

Training model is the same as above.

Training Algorithm

Load train data and test data from tfrecords.

Initialize training hyperparameters.

for epoch = 1, E **do**

  for iter = 0, I_T **do**

    Generate random batch from train data

    Do FP & BP, compute loss.

    Optimization: SGD with exponential decay

  end for

  for iter = 0, I_V **do**

    Generate sequential batch from test data

    Compute average accuracy

  end for

  if average accuracy > best accuracy **do**

    Best accuracy = average accuracy

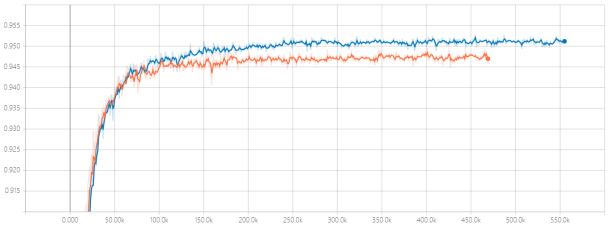
    Save model

end for

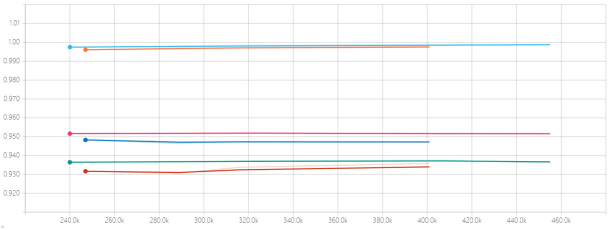
---

5. Results

5.1. Project Results



Name	Smoothed	Value	Step	Time	Relative
No_Wave_val	0.9470	0.9460	469.0k	Tue Dec 11, 17:58:11	14h 37m 45s
Wave_val	0.9512	0.9514	555.0k	Wed Dec 12, 17:41:51	16h 51m 21s



Name	Smoothed	Value	Step	Time	Relative
No_Wave_eval\test	0.9340	0.9358	401.0k	Wed Dec 12, 18:01:47	40s
No_Wave_eval\train	0.9975	0.9981	401.0k	Wed Dec 12, 17:59:18	9m 16s
No_Wave_eval\val	0.9472	0.9472	401.0k	Wed Dec 12, 18:00:52	1m 7s
Wave_eval\test	0.9367	0.9361	455.0k	Wed Dec 12, 18:42:17	40s
Wave_eval\train	0.9987	0.9990	455.0k	Wed Dec 12, 18:39:51	9m 14s
Wave_eval\val	0.9516	0.9515	455.0k	Wed Dec 12, 18:41:22	1m 8s

As shown in the figures, the model with wavelet transform is able to achieve a validation accuracy of 95.14%.

5.2. Comparison of Results

As the figures shown above, the validation accuracy of the model with wavelet transform is 95.14% while the original model achieves a validation accuracy of 94.70%,

which means by introducing additional image preprocessing methods, the accuracy is improved.

The training speed of model with wavelet transform is 15.28% slower than the original model without preprocessing. At the meantime, the accuracy is improved. Although preprocess may increase the training time, the result is promising. It's natural to be more time consuming when preprocess steps are applied and that's why the object of convolutional neural network is to use raw images.

5.3. Discussion of Insights Gained

During our experiment, if we use less data, the final accuracy will get worse and the model will get easier to converge, which means a 11-layer neural network is of too much capacity and overfitting may occur if the data set is not large enough. Consequently, the main task is to increase the diversity of the inputs. Intuitively, we can try to collect more training data. But this is really time consuming and not applicable. On the other hand, we can try to explore more information from the current available dataset and reduce the information redundancy.

From the plotted result, we can conclude that the accuracy is higher after the image preprocessing method is adopted. As mentioned in the algorithm introduction, Fourier transform and wavelet transform can provide more information from both the spatial and spectral domain. Besides, as images are transformed from the original RGB form to the grayscale form, much information redundancy from the spatial domain is avoided.

Our result comparison demonstrates that our analysis to some degree improves the performance of the same model. For the original model, writers provided a method that only use chopping and reshaping to generate input images which results in a good accuracy. In the process of modifying different hyperparameters, when the kernel size is decreased from 5 to 3, the speed of training is improved while the accuracy will decrease. By using hyperparameters provided by the original writers, we are able to achieve a high degree of validation accuracy.

In the original paper, because sequence with 6 or more than 6 digits is rare, the writer didn't take that into consideration. In the training process, when we took that situation into consideration, the validation accuracy is decreased to 82.41% which is not satisfying. Thus, we had a discussion about the real situation. In the training data where there are thousands of images, only one sequence of 6 digit is provided. If the number of sample provided is much larger, the accuracy while taking 6 digits into consideration will be increased significantly.

6. Conclusion

In our project, we firstly rebuild the model in the original paper with accuracy of 94.60%, and then adopt a new preprocessing method to extract the information from spectral domain. Finally, the validation accuracy is improved to 95.14%. According to our result, we realize the power of convolution neural network but the hyperparameter of kernel limits the ability of feature extraction from the dataset. For further development, we hope to take the spectral domain into consideration and apply other method like GCN to improve the accuracy of digit recognition.

6. Acknowledgement

We would like to thank Chenyang Lei for helpful discussions.

7. References

Include all references - papers, code, links, books.

[1] https://bitbucket.org/ecbm4040/2018_assignment2_yd2459/src/master/

[2] Inc, Google . "Convolutional, long short-term memory, fully connected deep neural networks." IEEE International Conference on Acoustics IEEE, 2015.

[3] 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*.

[4] Goodfellow, Ian J., et al. "Multi-digit number recognition from street view imagery using deep convolutional neural networks." arXiv preprint arXiv:1312.6082 (2013).

8. Appendix

8.1 Individual student contributions in fractions - table

	YS3127	YD2459	JM4743
Last Name	Sun	Ding	MA

Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	Implemented the model	Implemented the training algorithm	Rebuild the TFrecord transformation
What I did 2	Test the model with different kernel size and padding laws.	Implemented the code to download, extract the dataset, and count its basic information	Introduce wavelet transform into image processing and the model input
What I did 3	Train the network with different hyperparameters	Train the network with different hyperparameters	Improve the accuracy slightly