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

E4040.2019Fall.RNDM.report
Mingfei Sun ms5898, Yifeng Deng yd2505, Fan Wu fw2322
Columbia University

Abstract

We use CNN models to recognize arbitrary multi-digit numbers from Street View imaginary. The main technical challenge is to determine the number of parameters and the depth of models for better testing accuracy. We find that depth of CNN models is necessary for good performance, and large shallow models cannot achieve the same results.

1. Introduction

Recognizingthe multi-digit numbers in photographs is an important part of modern-day map making. Google's Street View imagery comprised of many gep-located 360 degrees images.The method to automatically recognize an address number from a photograph helps pinpoint the location of the building with a high accuracy. Because of the variability in the visual appearance of these address numbers in the wild with much colors,styles, orientations and so on, this task is difficult. Furthermore, the task is complicated by environmental factors such as lighting and shadows and by image acquisition factors such as resolution and focus blurs.

Because of these complexities, traditional approaches to the task separate out the localization,segment and recognitionsteps. In our project, we mainly refer to the paper[2]. We used a unified approach, discussed in the paper, that integrates localization, segmentation and recognition steps via deep convolutionalneural networks(CNN) to do the task. We evaluated our models on publicly available Street View House Numbers(SVHN) dataset and achieved 80.29% in testing accuracy.We also trained different models from shallow models to deep models to compare the performance of these models and we also changed the number of parameters in a model to compare the performance.

One of the main difficulties are training different shallow or deep CNN models with different parameter to compare the results. Because of limited resource,we tested several models to get the bestperformance of models with different layers.Another difficulty is that it took a long time to run the code, two weeks in the original paper, but we only had to train a model in six hours, taking into account the limitations of computing resources.

2. Summary of the Original Paper

In section 2.1,we discuss the special part of the address number recognition task of the originalpaper, and some parts of models and code base on this part. In section 2.2 we would show the main results of the paper.

2.1 Methodology of the Original Paper

The original paper tells anunified approach that integrates localization,segmentation and recognition steps via deep CNN to recognize the address numbers in images. And street number transcription is a special kind of sequence recognition. Given an image, the task isto identify the number in the image and the length of these numbers.

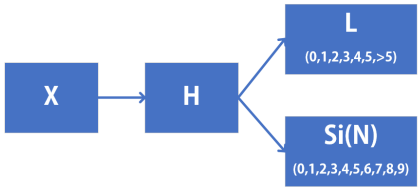


Fig 1. The block diagrams of the learning network (11 layers)

From Fig 1,X is the input images, H is the features getfrom X, L represents the length of the address number in an image, and L could be 0,1,2,3,4,5 and "more than 5". Si represent each number in the address number sequence, and the number has 10 possible values.Thus the author uses softmax models for each variables(L, and every Si).

2.2 Key Results of the Original Paper

The main results of the original paper could be represents by the following figures.

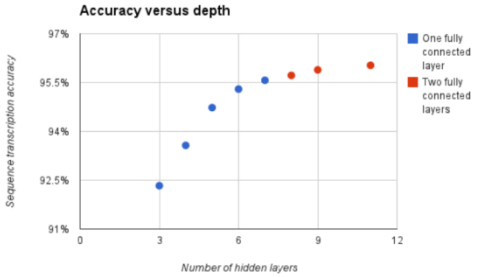


Fig 2 sequence transcription accuracy based on architectures

Fig 2 shows that fairly deep architectures are needed to obtain good performance on the task. 11 layers model has the best performance and the sequence transcription accuracy achieved 96.03%. Though a model with more

hidden layers than 11 could get a little better accuracy, the cost of training and testing could increase extremely.

The best model of the paper shows as below: the number of units at each spatial location in each layer is [48, 64, 128,160] for the first four layers and 192 for all other locally connected layers. The fully connectedlayers contain 3,072 units each. Each convolutional layer includes max pooling and subtractivenormalization. The max pooling window size is 2 × 2. The stride alternates between 2 and 1 at eachlayer. All convolutionsuse zero padding on the input to preserve representation size. All convolution kernels were of size 5 × 5.

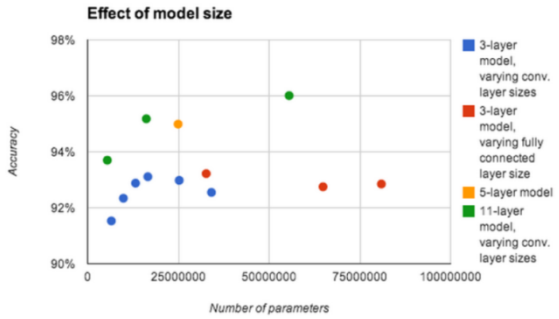


Fig 3 sequence transcription accuracy based on numbers of parameters

Fig 3 shows that increasing thenumber of parameters in smaller models does not allow such models to reach the same level ofperformance as deep models, primarily because of overfitting.

3. Methodology

Section 3.1 shows main objectives and challenges of our project. The section tells about our motivation, tasks,objectives and challenges. Section 3.2 shows the Problem Formulation and Design.

3.1. Objectives and Technical Challenges

We used the same models mentioned above in section 2.2 to vary the results of the paperand designed our own code to complete the address numberrecognition task. The paper didnot tell the parameter of other CNN models except 11 layers model, so we designed other models such as 3-layer model, 4-layer model and so on and trained these models many times to find the fairly good parameters. Because Fig 3 only shows the results of a part of models, we trained different models to prove the result showed by Fig3.

We met some challenges : 1 We don't have a photo library other than SVHN; 2 The hyper-parameters in the paper are not completely given; 3 Computing resources are limited.

3.2. Problem Formulation and Design

Our basic approach is to train a probabilistic model of many given images. If **S**represent the output sequence of numbers and **X**represent the input image. Our goal is to learn a model of P(**S** |X) by maximizing log P(**S**|X) on the training set.

Now we define: **S** represents N random variables $S_1,S_2,\ldots\ldots,S_N$, and L represents the length of the address number.If the identities of these variables are independent from each other, the peobabilit of a specific sequence **s** $=s_1,s_2,\ldots\ldots,s_n$ is given by:

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

In this case, L could be 0,1,2,3,4 and 5.If we want L represents some numbers bigger than 5,we could define a number to represent the case. s_i has ten possible values such as 0,1,……,9. Fig 1 shows the relationship between these variables. Therefore, we could predict **S**,when we get:

$$\text{argmax}_{L,S1,\ldots,Sn}\log P(\mathbf{S}\ |X)$$

Based on these formulas, we design our own CNN models to classify these variables and then recognize the address number of the images.

4. Implementation

In this section, we will demonstrate our implementation in two parts. The first is the description of the learning network we build ourselves after learning from the original paper. Secondly, we will show the design of the whole processing system of our project step by step.

4.1. Deep Learning Network

1) The structure

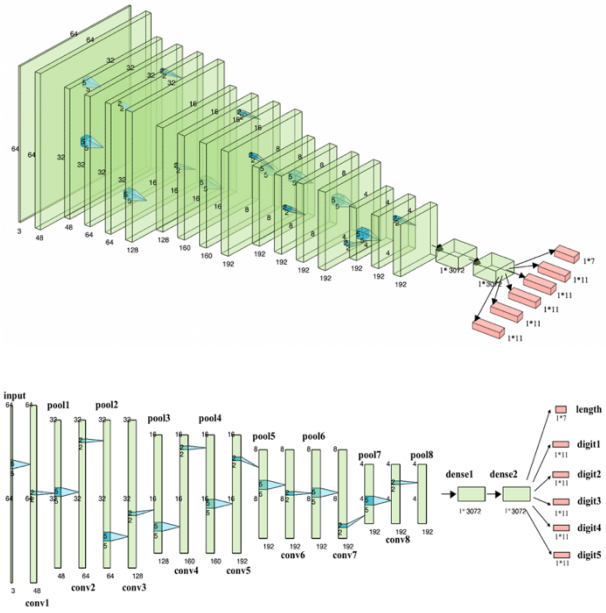


Fig 4. The block diagrams of the learning network (11 layers)

As shown in Fig 4, we build the 11-layer network consisting of eight convolution hidden layers and two fully connected hidden layers (plus one output layer) with the size parameters described in the original paper which is said to be tested achieving the best performance. The convolution layers' sizes are 48, 64, 128, 160, 192, 192, 192, 192 with kernel size of 5*5. The max pooling kernel size is 2*2 for all and takes stride as 2 every 2 pool layers (like 2, 1, 2, 1,…). In our implementation, the best result we got is 80% test accuracy under this model.

2) *Training details*

In consideration of time consumption, we trained the model with 30720 training examples encapsulated in batches with size of 32 (one round consisting of 960 batches). The whole training process last for 10 epochs (repeated for ten times). Regarding validation, we test 100 out of 2680 validation examples every 300 batches. Apart from that, we employed the Early Stopping in our training algorithm as regularization to prevent overfitting.

3) *Dataset*

In our project, we chose the Street View House Numbers (SVHN) dataset [2]as the material we apply our project on. SVHN is a dataset containing 200,000 street number images obtained in Google Street View. Each individual digit in the image is bounded by pre-set bounding box whose location and size are recorded.

4.2. Software Design

1) *System flowchart*

As is pictured below, the while system of our project contains three main steps: data pre-processing, model training and prediction.

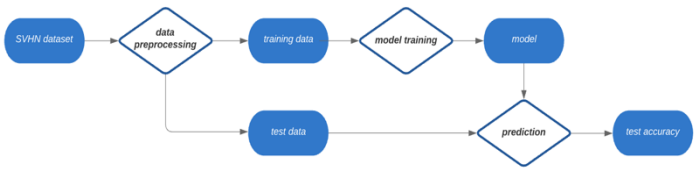


Fig 5. The whole processing flowchart

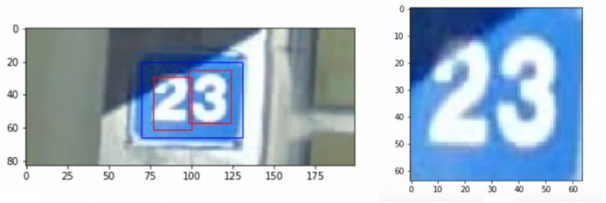
2) *Data pre-processing*

Asis mentioned before in 4.1 (3), the images in the dataset have each individual digit bounded out for digit recognition training. In our program, we aim to use one network recognize multi-digit sequence, which means that we need the images of bounded sequences instead of independent digits for training. Therefore, some pre-processing work should be done to satisfy our needs.

Fig 6. The flowchart of data preprocessing

The data pre-processing's flowchart is shown in Fig 6.above. The pre-processing algorithm goes as follows: first we collect all the digits' location information in one image; then calculate the left side (the most left coordinate), the top side (the highest coordinate), the max width and the max height as the sequence boundary; after that, expand the bounding box we get before by 30% (each direction 15% and remember to consider the potential case of overflow); at last, crop the image within the bounding box and resize it to 64*64. This process should also be applied to the test dataset. Fig 7.shows one example of the data pre-processing below.

Fig 7. (a)The original image: the red squares are digit bounding boxes and the blue square is generalized sequence



bounding box; (b)the final image after pre-processing (resized).

3) *Training and prediction*

The training details referring to the learning network have been discussed before in 4.1 (2). After the whole training process, we record the model by saving it with TensorFlow. Training process is the most time-consuming part throughout the whole project.

As for prediction, we first read and import the model we saved after training process withTensorFlow, apply the weights to the test examples to get predictions, and compare them with known labels to calculate test accuracy.

5. Results

5.1. Project Results

1) *Best result*

Test accuracy:80.29%
Parameters:
Conv1: 96
*Conv2:*128
*Conv3:*256
*Conv4:*320
*Conv5:*384
*Conv6:*384
*Conv7:*384
*Conv8:*384
*Dense1:*6144
*Dense2:*3072
Training time:6 hours
Verification accuracy:84%

2) *Performance variations on differentnetwork depths*

As we can see from Fig 8., the deeper the network is, the better the performance we can get. Also, the improvements in performance by deepening the network is significant.

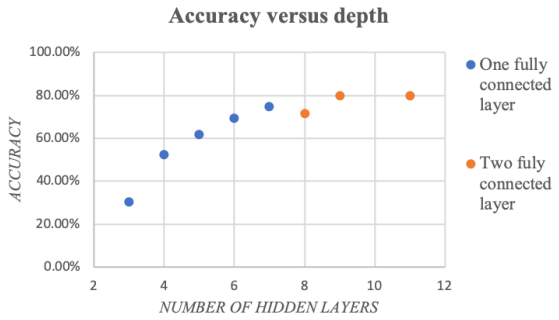
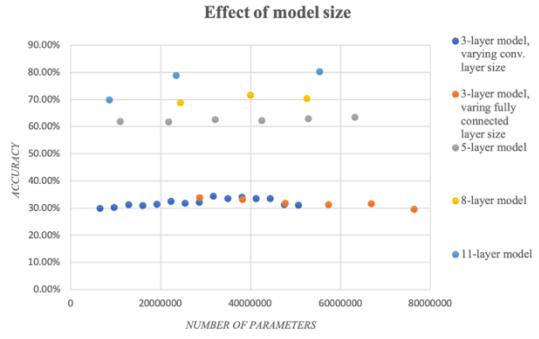


Fig 8. Test accuracy with different model layers

3) *Performance variations on different parameter sizes*

As we can see from Fig 9., the parameter size has little influence over performance within models of same depth, compared with variations over depth of the network.

Fig 9. Performance analysis experiments over parameter size



with different layer-depth

5.2. Comparison of Results

1) *Comparison of the best result*

	Test accuracy	Training time	Network depth	Hardware
Original paper	96.03%	6 days	11 layers	10 replicas in DistBelief
Our project	80.29%	6 hours	11 layers	1 x NVIDIA Tesla K80, 2CPU

Table 1. Detail comparisons between original papers and this project

2) *Comparison of the performances over models of different depths*

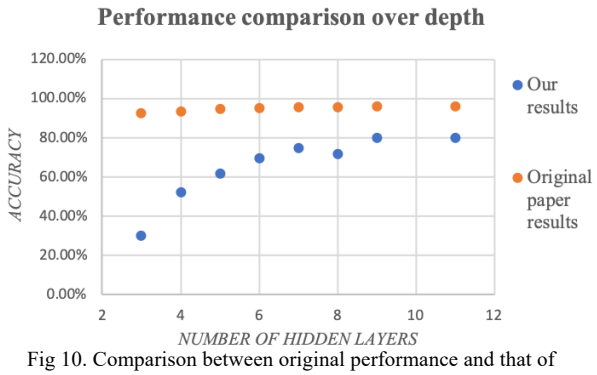


Fig 10. Comparison between original performance and that of this project over network depth

5.3. Discussion of Insights Gained

Cause of deficiency:

1. limitation of the computing power: The origin paper trained the model took approximately six days using 10 replicas in DistBelief. But we only have 1 NVIDIA Tesla k80 GPU, 2 CPUs and 6 hours to train the model
2. Hyper-parameter: Hyper-parameter is not given in the original paper, so we have to do Hyper-parameter tuning by ourselves. The number of network layers used in this project is large so the hyper-parameters need to tune is also large, it is difficult to achieve the accuracy of the original paper.
3. Quality of Some data in data sets are poor: Some of the images in the dataset are small in size. When the images are zoomed to 64x64 and then cut down to 54x54, the contents of the images cannot be recognized easily.

Fig 11. Poor quality of input examples

Efforts to increase accuracy:

1. Use Batch Normalization in the network: BN makes the model less sensitive to the parameters in the network, simplifies the tuning process, and makes the network learning more stable. Because the network is relatively deep, we used BN.
2. Use data augmentation: SVHN dataset have over 10000 images, with data augmentation we get over 20000 images for training set.
3. Change the number of parameters of the network: We used neural networks with the same number of layers but different Numbers of parameters for comparison. In the 11-layer neural network, increasing the number of parameters increased accuracy by 1%
4. Try different layers of neural networks: We tried to change the number of convolutional and fully connected layers in the neural network and compared their accuracy. Finally, we determined that the 11-layer neural network had the highest accuracy

6. Conclusion

In this project, we learn to rebuild a learning network for multi-digit recognition, compare the performance of our own model with previous results and implement performance experiments over both model size and model depth. Our best result achieved in test accuracy is 80.3%, which is lower than that of the original paper. The deficiency mainly lies in lack of training time and training examples. For future work, apart from prolonging training time, we plan to employ techniques of Batch Normalization, data augmentation and hyper-parameter tuning to improve the performance as well.

7. References

[1]https://github.com/cu-zk-courses-org/e4040-2019fall-project-rndm-ms5898-yd2505-fw2322.git

[2] Goodfellow, Ian &Bulatov, Yaroslav&Ibarz, Julian &Arnoud, Sacha &Shet, Vinay. (2013). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.

[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*.

8. Appendix

8.1Individual student contributions infractions - table

	CUID1	CUID2	CUID3
Last Name	Mingfei Sun	Yifeng Deng	Fan Wu
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	Preprocess data	Build 6,7,8 layers network	Build 3,4,5 layers network
What I did 2	Data augment	Part1 of the report	Part of the report
What I did 3	Build 11 layers and 9 layers network		