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

ECBM4040.2017Fall.report
Qing Ma qm2124, Nanbin Xia nx2138
Columbia University

Abstract

Recognizing arbitrary multi-character text in unconstrained natural photographs is a hard problem. In this paper, we address an equally hard sub-problem in this domain viz. recognizing arbitrary multi-digit numbers from Street View imagery. Traditional approaches to solve this problem typically separate out the localization, segmentation, and recognition steps. The original paper proposes a unified approach that integrates these three steps via the use of a deep convolutional neural network that operates directly on the image pixels. We apply this method and realize recognizing multi-digit number from street view images.ance occurring in the deepest architecture we trained, with eleven hidden layers. We evaluate this approach on the publicly available SVHN dataset and achieve over 95% accuracy in recognizing complete street numbers.

1. Introduction

Recognizing multi-digit numbers in photographs captured at street level is an important component of modern-day map making. An example of a corpus of such street level photographs is Google's Street View imagery comprised of hundreds of millions of geo-located 360 degree panoramic images. The ability to automatically transcribe an address number from a geo-located patch of pixels and associate the transcribed number with a known street address helps pinpoint, with a high degree of accuracy, the location of the building it represents. The recognition problem is further complicated by environmental factors such as lighting, shadows, specularities, and occlusions as well as by image acquisition factors such as resolution, motion, and focus blurs.

In this problem, we use deep convolutional neural network that operates directly on the image pixels. This model is configured with multiple hidden layers, all with feedforward connections. We employ DistBelief to implement these large-scale deep neural networks. We have evaluated this approach on the publicly available Street View House Numbers (SVHN) dataset and achieve over 95% accuracy in recognizing street numbers.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

Because our project mainly focuses on retrieving the results according to the reference paper and further modification on the model, parameters and optimizers, the methodology of paper is close to that of the original paper. Please refer to Section 3. Methdology.

2.2 Key Results of the Original Paper

The key results of this paper are: (a) a unified model to localize, segment, and recognize multi-digit numbers from street level photographs. (b) a new kind of output layer, providing a conditional probabilistic model of sequences. (c) empirical results that show this model performing best with a deep architecture.

The best model obtained a sequence transcription accuracy of 96.03% while using confidence thresholding we obtain 95.64% coverage at 98% accuracy.

3. Methodology

For Image Preprocessing, We first find the small rectangular bounding box that will contain individual character bounding boxes. We then expand this bounding box by 30% in both the x and the y direction, crop the image to that bounding box and resize the crop to 64 × 64 pixels. We then crop a 54 × 54 pixel image from a random location within the images, and flattened the 54*54*3 matrix to a sequence of length 8748. This sequence serves as the inputs X.

We feed the preprocessed data to the convolutional neural network. Since very few street numbers contain more than five digits, so we can use models that assume the sequence length n is at most some constant N, with N = 5 for this work.Also, there is no "partial credit" for getting individual digits of the sequence correct.

Our goal is to learn a model of $P(S \mid X)$ by maximizing the log of posterior $\log P(S \mid X)$ with application of convolutional neural networks on the training set. It can also be used to detect if the assumption that the length of the sequence is at most N.

3.1. Objectives and Technical Challenges

Our objective is to use Deep Convolutional Neural Networks to recognize street number images from Street View House Numbers (SVHN) dataset. We need to

separate the object into multiple steps and get over technical challenges of each step.

The very challenging part in data preprocessing is to load pixel information from .png or .jpeg file and converting it info forms of data that could be utilized for our Convolutional Neural Networks model. Because we are unfamiliar with the image processing, we used a lot of Python scripts to explore the preprocessing results.

The modeling process is the most challenging part of the project. Realizing the model and fit into the data is complicated and time-consuming. Modeling guidance from our reference paper [2] helps me a great deal.

3.2. Problem Formulation and Design

3.2.1. Formulations

As mentioned at the start of this section, our goal is to maximizing log P(**S** | X) on the training set while **S** stands for the output sequence and X represents the input image. **S** consists of variable L and sequences of variables S1, S2, …, Sn. L represents the length of the sequence and the sequence S represents its elements.

We assume that the identities of the separate digits are independent from each other, so that probability of a specific sequence s = s1, s2, …, sn is given by:

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

At test time, the sequence we predict based on

$$\mathbf{s} = (l, s_1, \ldots, s_l) = \text{argmax}_{L,S_1,\ldots,S_L} \log P(S \mid X).$$

The calculation of argmax can be done in linear time independently. Thus, by adding up the linear time, the total running time is O(N)

3.2.2. Discussion of Our Software Design

Software Design	
Literature Review	
Data Extraction	
Data Prepocessing	
Model Design	
Training Data	Building Layer
Evaluation	
Visualization	

Table 1: Software Design Workflow

Table 1 shows the discussion of our software design. We design our software by separating the objective into multiple steps and discuss the methods of how to get over each step.

For part of Data Preprocessing, Training Data and Visualization, we write the code into the Jupyter Notebook file since the process of these part need checking step by step. The rest of the part are written in Python functions for repetitive use and easy modification.

4. Implementation

Based on the results of accuracy from our reference paper [2], we first implemented our Convolutional Neural Network (CNN) model consisting of eight convolutionary hidden layers, one locally connected hidden layer and two densely connected hidden layers. All connections are feedforward and go from one layer to the next. The detailed construction of each hidden layer and training algorithms will be introduced in Section 4.1.2.

4.1. Deep Learning Network

4.1.1. Architectural Block Diagrams

Figure 1 shows the general implementation of our CNN model containing hidden layers and connections..

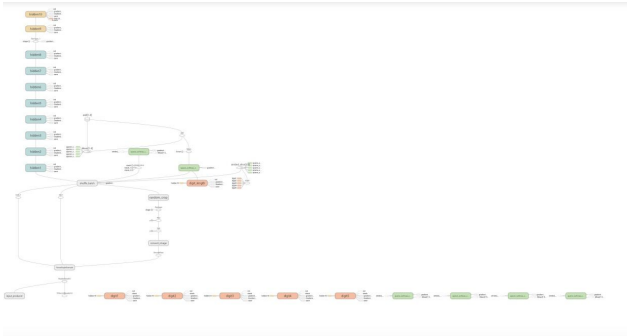

Figure 1: CNN Diagrams

4.1.2. Training Algorithms

Among all eight convolutionary hidden layers, one locally connected hidden layer and two densely connected hidden layers, the first hidden layer contains maxout units[3] while the others contains rectifier units[4]. 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 the other locally connected layers. The fully connected layers contain 4 * 4 * 192 = 3072 units each. The max pooling window size is 2 * 2 and the stride alternates between 2 and 1 at each layer, so that half of the layers do not reduce the spatial size of the representation[1]. The entire CNN inference process is shown below in the figure.

CNN Inference	
Layer	**Description**
Input Layer	Input: Image array
Conv2d - 1	Filter: 48, Kernel: 5 * 5, Padding: Valid
ReLU	Rectified Linear Unit Activation
Max Pooling	Pool: 2*2, Strides: 2, Padding: Valid
Conv2d - 2	Filter: 64, Kernel: 2 * 2, Padding: Valid
ReLU	Rectified Linear Unit Activation
Max Pooling	Pool: 2 * 2, Strides: 1, Padding: Valid
...	...
Conv2d - 8	Filter: 192, Kernel: 5 * 5, Padding: Valid
ReLU	Rectified Linear Unit Activation
Max Pooling	Pool: 2 * 2, Strides: 1, Padding: Valid
Flatten	Reshape: [-1, 3072]
Dense Layer - 1	Units: 3072
Dense Layer - 2	Units: 3072
Dropout	Length: L, Digits: S1, S2, ..., S5

Table 2: CNN Inference

During the training process, we choose 0.01 as the learning rate and 0.9 as the decay rate. In the batch size of 32, we build batch for the image, the length of the elements mentioned in 3.2.1, and the sequence of digits. We use the total loss of cross entropy as the minimizer. We choose the Stochastic Gradient Descent method as the Tensorflow optimizer at first to retrieve the result of our reference paper, and then we use Adam Optimization method to control the learning rate in order to achieve better result.

4.1.3. Dataset

The dataset used in this project is the Street View House Numbers (SVHN) Dataset. SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images. The original paper used the same dataset.

4.2. Software Design

Because our project mainly focuses on retrieving the results according to the reference paper and further modification on the model, parameters and optimizers. Software design is relatively direct and clear based on the algorithm description. The Step-by-step implementation of algorithms is mentioned in Section 4.1.2 and the overall structural design is mentioned in Section 3.2.2.

5. Results

5.1. Project Results

The results we get from this project include the trained model with best accuracy of 95.0%, the loss of every 100 steps and the accuracy evaluated every 1000 steps.

We apply “patience” parameter as stop condition, the stop condition is that no better accuracy is achieved while a specific number of steps after best results.

The best result is given by model with 11 layers with patience 50 is at 214500 training steps, the time consumed is 12 hours 10 minutes.

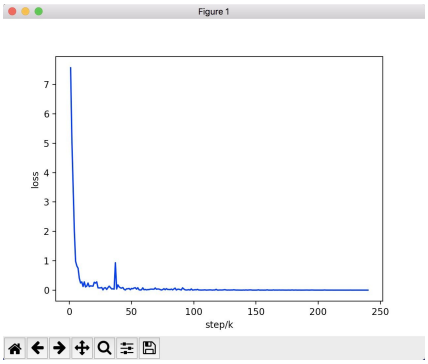

Figure 2: loss versus steps

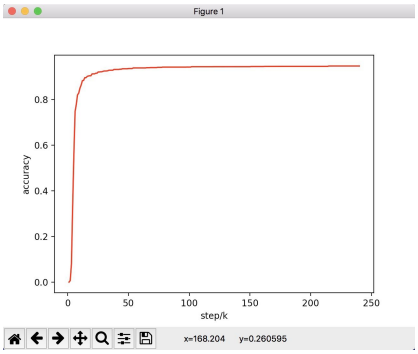

Figure 3: Accuracy versus steps of network with 11 layers

Test Result

To evaluate the result, we load the trained model and test it with test images.

Figure 4 shows the result, after the image is give, the program predict the digits and the length. Digit "10" means no digits

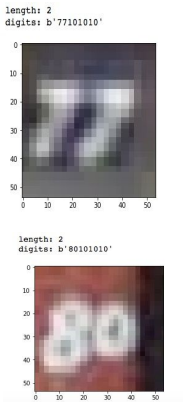

Figure 4: Prediction on test image.

5.2. Comparison of Results

Model with different hidden layers

In the project, we build the networks of different number of convolutional layers, the last two layers of each model are fully connected layers with relu activation.

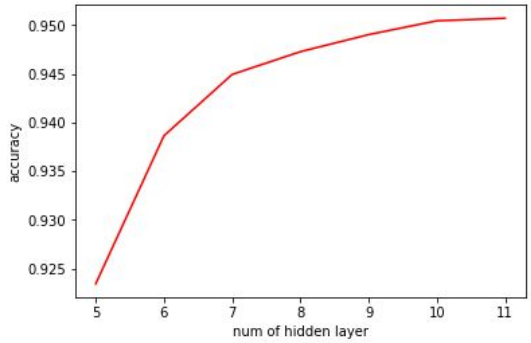

Figure 5: Accuracy versus number of hidden layers in our project

In the original paper, the networks with 3 to 7 layers do not have fully connected layer.

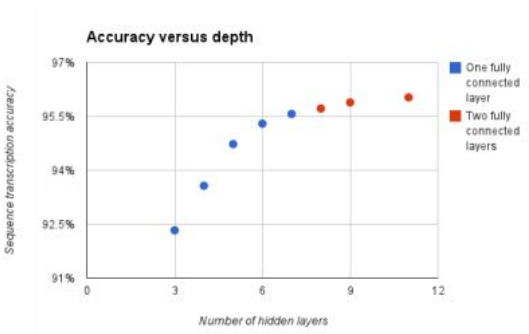

Figure 6:: Accuracy versus number of hidden layers in the original paper

From the figure we could say that our project and the original paper yield similar result. As the number of hidden layers increase, the accuracy goes up. In the paper, networks achieves the best result when the layer number is eight while in our project the accuracy keeps going up. The increasing rate may depends on the choice of different type of layers, activation functions as well as optimization method.

In our project the best accuracy is 96.1% after running 280000 steps while in the original the number is 96% before applying confidence thresholding.

The original paper get 98.7% accuracy after discarding pictures with low confidence. The total degree of accuracy of both system remains the same

5.3. Discussion of Insights Gained

Generally, the number of hidden layers increase, the accuracy will goes up. However in the paper, the best result is achieved while using networks with 8 hidden layers while in our project it keeps going up. We assume that the increasing rate or the best architecture may depends on the choice of different type of layers, activation functions as well as optimization method.

In this project we use adam optimizer and gradient descent optimizer on all trained networks, the results given by adam are better in each case.

Both our project and the paper preprocess by subtracting the mean of each image. In the further implementation, we could apply local contrast normalization.

6. Conclusion

In this project, we achieved the goal of unifying an approach that integrates the steps of localization, segmentation, and recognition via the use of a deep convolutional neural network that operates directly on the image pixels.

Through this project, we accumulated a lot of experience on implement tensorflow and working with GPU.

Our results with this architecture is based on the assumption that the sequence is of bounded length, with a reasonably small maximum length N. For unbounded N, our method is not directly applicable, and for large N our method is unlikely to scale well.

7. Acknowledgement

We would like to thank Professor Kostic for providing us so many resources. We would also like to thank TAs for providing helpful discussions.

8. References

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

[1]

[2] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. *Multi-digit Number Recognition from Street View Imagery Using Deep Convolutional Neural Networks*, 2013

[3] Ian J. Goodfellow, Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. *Maxout Network*, 2013
[4] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. *What Is The Best Multi-stage Architecture for Object Recognition?* 2009

9. Appendix

9.1 Individual student contributions - table

	qm2124	nx2138	
Last Name	Ma	Xia	
Fraction of (useful) total contribution	1/2	1/2	
What I did 1	Code Implementation	Code Implementation and Test	
What I did 2	Report Writing	Report Writing	
What I did 3			