Cap2TxT: CAPTCHA Image to Text Sequence, An End-to-End Hybrid Neural Network for Captcha Image Text Sequence Recognition

2020 Spring ML Class Final Project Submission

Changwoon Choi, 2014-17733
Electrical and Computer Engineering, Seoul National University
Seoul 08826, South Korea
zzzmaster@snu.ac.kr

Abstract

Recent developments of deep neural networks including CNN(Convolutional Neural Network) and RNN(Recurrent Neural Network) made object classification and detection processes much more easier. However, many real-world sequence learning tasks require the prediction of sequences of labels from noisy, unsegmented input data. In this paper, I propose ***Cap2TxT****, a light-weight end-to-end fashion network for captcha image recognition problems. Codes are available at author's github repository.*[1]

1. Introduction

Recently, in light of the great success in deep learning networks, computer's pattern recognition capabilities have been remarkably improved. Especially, the pattern recognition for image in the computer vision field started to be in the limelight. However, most of the recent deep neural networks are focusing on solving object classification and semantic/instance segmentation problem. This project deals with the problem of image-based text sequence detection, a problem that has been covered a lot in the field of computer vision in the past but rarely addressed from a deep learning approach. In particular, we limit the problem domain to the CAPTCHA image consisting only of English alphabets and arabic numbers. I coined my network ***Cap2TxT*** for Captcha Image to Text Sequence.

Unlike image classification in which deep learning and CNN are most actively used, recognizing objects that inherently have the properties of sequence(*e.g.* text information in CAPTCHA, the musical score) has some challenges. Those objects that have the feature of sequence can't be handled as image classification problem since the number of labels or classes are infinity. Thus, it is more natural to predict the order or sequence of labels rather than to give a single label. Also, an important difference between objects with ordering properties is that the length of the sequence can vary significantly.(*e.g.* ”Bob” versus ”Josephine”) Therefore, famous existing successful CNN networks such as AlexNet[7], GoogLeNet[15], ResNet[4] and DenseNet[5] cannot be applied directly to this area without modification. So in this work I leveraged RNN to handle those sequence-based objects. The Cap2TxT framework extracts feature sequences through CNN as the first step. Then, the text sequence is predicted for each time step(the time sequence in the RNN is the horizontal sequence of input image)through the RNN. Finally the loss function between per-sequence-predicted results and ground truth texts(which are unsegmented at sequence level) is computed by the method inspired by [3]. I will describe the details of network architecture later in this paper in section 3.

1.1. Related Works

In this subsection, I will give a brief overview of the research area related to my work including OCR and deep neural networks.

OCR(Optical Character Recognition) is the automated translation of images of typed, printed or handwritten text into coded text, whether from a scanned document, a photo of a document, from subtitle text overlying on an image.[10] OCR enables a large number of useful applications[6]. During the early days, OCR has been used for mail sorting, bank cheque reading and signature verification[14]. In addition, there are a lot of applications of OCR such as helping blind and visually impaired people to read text[1], processing utility bills, passport validation, pen computing and automated number plate recognition[11].

Before the deep neural network was used for OCR, the

[1]https://github.com/changwoonchoi/mlfinal

classical pipeline of OCR was as follows: 1) Image acquisition, 2) Preprocessing, 3) Character segmentation, 4) Feature Extraction, 5) Classification, 6) Post-processing. As you can see, the classical OCR was very cumbersome and computationally and time consuming.

Deep Neural Networks. Recent years, due to the rapid development of the hardware such as GPU(Graphics Processing Unit), the study of deep neural networks has flourished. But the success in deep neural network is not just the result of powerful hardwares or larger datasets, but is mainly a result of new algorithms and novel network architectures. By LeCun *et al*. [9],which was a pretty simple model, CNN was first used for the image classification task. Nowadays, deeper and more powerful models have been studied through AlexNet[7], GoogLeNet[15], ResNet[4] and DenseNet[5].

If CNN has already had great results in image classification, deep neural networks have been very successful with the advent of RNN in areas such as NLP(Natural Language processing). RNN was first introduced in [13]. But as the network grew deeper, problems such as banishing gradients arose. [8] introduced LSTM(Long Short-Term Memory) to deal with those problems, and produced remarkable results. Therewithal, [2] presented GRU(Gated Recurrent Unit), which also achieved a success in sequence predicting.

1.2. Main Contribution

The main contribution of this paper is three-fold. In summary, the contributions are as follows:

- The neural network model I proposed doesn't put leverage on certain characteristics of the given character.(*e.g*. English alphabet, arabic numeral) So Cap2TxT can be trained to recognize the alphabet of all kinds of languages. In theory, it can even learn to recognize non-linguistic image data like musical notes.
- Since the proposed network does not utilize any dictionary vocabulary information during the training/learning process, Cap2TxT can flexibly cope with any string(or text sequence) even if it is a nonsense word. That is, it is very suitable for CAPTCHA recognition.
- Having an end-to-end fashioned pipeline, it is easy to train network compared to other models for OCR that consist of multiple processes chunks.
- Owing to the architecture of proposed network, it handles image containing text sequence of various lengths.

The rest of this paper is organized as follows. Section 2 gives a detailed description of the *Cap2TxT* network architecture that I proposed in this final project. In Section 3, I describe the experimental results and methods, and introduce several candidate models for *Cap2TxT* network that have undergone trial and error during the project, and Section 4 concludes this paper.

2. Proposed Network Architecture

The whole *Cap2TxT* network architecture is shown in Figure 1. The network consists of three parts: 1) CNN layers 2) RNN layers 3) Measuring Loss Value Step.

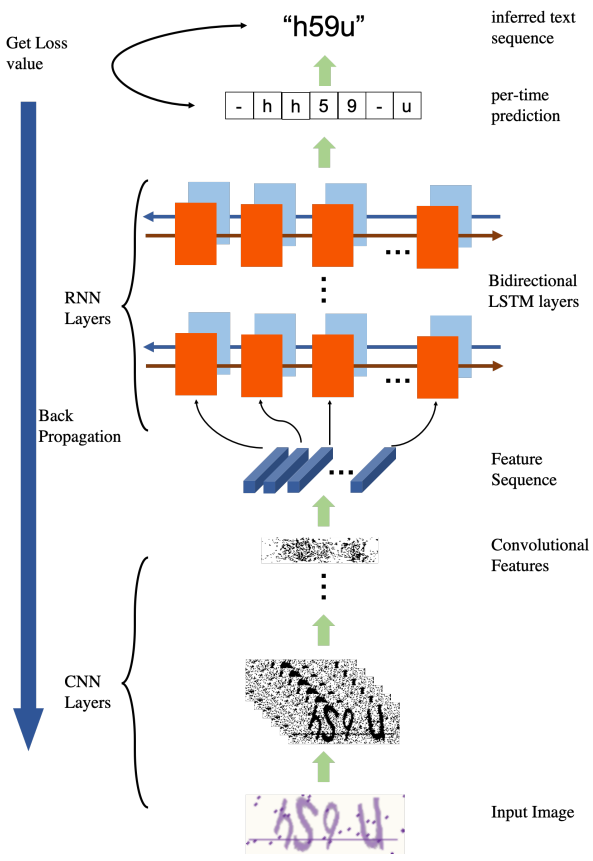

Figure 1: The overview of Cap2Txt architecture. The network is composed of two parts: 1) CNN layers: It takes captcha image as input and extracts a feature sequence. 2) RNN layers: It takes a feature sequence for input and returns per time predicted character labels. 3) Measuring Loss value step : from per time prediction, it transcripts and calculates the loss between the ground truth text and the inferred text sequence.

2.1. CNN Layers

The first part of the proposed network is a convolutional layer block. Inspired by [15] as assignment3, CNN layers consist of deep-layered inception blocks, which contains bottleneck layers, 1x1 convolution layers, to reduce

type	patch size/ stride/ padding	output size (WxHxc)	depth	# 1X1	# 3X3 reduce	# 3X3	# 5X5 reduce	# 5X5	pool proj
convolution	7x7/1/3	160x64x64	1						
ReLU			0						
max pool	3x3/2/1	80x32x64	0						
convolution	1x1/1/0	80x32x64	1						
ReLU			0						
convolution	3x3/1/1	80x32x192	1						
ReLU			0						
max pool	3x3/2/1	40x16x192	0						
inception		40x16x256	2	64	96	128	16	32	32
inception		40x16x480	2	128	128	192	32	96	64
max pool	3x3/2/1	20x8x480	0						
inception		20x8x512	2	192	96	208	16	48	64
inception		20x8x512	2	160	112	224	24	64	64
inception		20x8x512	2	128	128	256	24	64	64
inception		20x8x528	2	112	144	288	32	64	64
inception		20x8x832	2	256	160	320	32	128	128
max pool	3x3/2/1	10x4x832	0						
inception		10x4x832	2	256	160	320	32	128	128
inception		10x4x1024	2	384	192	384	48	128	128
avg pool	4x4/1/0	**7x1x1024**	0						

Table 1: Network configuration summary of CNN Layers. The first row is input layer, and the last row is final output layer.

the dimensions and parameters remarkably. For easy and fast code implementation, I preprocessed the image data to make the height the same(64 pixels) with opencv library. Since the original CAPTCHA images have 60 pixels for height, I added extra black((r,g,b)=(0,0,0)) pixels at the bottom of images. Then the image passes through multiple layers composed of a number of convolution layers, ReLU activations, pooling layers and inception blocks. The Cap2TxT CNN configuration I used is summarized in Table 1. Note that the final output shape is (7x1xchannel#). I modified the CNN configuration to made the output shape as (Tx1xc) on purpose. From the fact that convolutional layers and pooling layers doesn't transform the position of pixels, making the output shape a (Tx1xc) form allows one to compress the image in the vertical direction(column-wise) and **preserve the text sequence information which was recorded in the horizontal direction**. Thus the final output extracted from the CNN layers implies the feature sequence in width-oriented time sequence domain. The core of concepts I explained is illustrated in Figure 2.

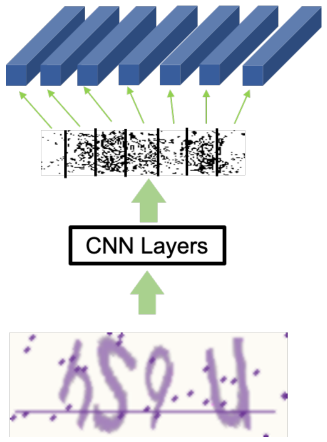

Figure 2: The CNN layers preserve the width-oriented time sequence of the original image. So it can leverage the original text sequence information in the image to extract feature sequence.

2.2. RNN layers

The extracted feature sequence from CNN layers is directly fed into RNN layers. A (Txc) shape can be considered as inputs of c-dimensional sequence of length T. I used a bidirectional LSTM model(,not the uni-directional model) which is implemented in pytorch library to boost performance. Also, by heuristic, I thought reducing (c=1024) dimension into (# of classes = 37) is too hasty, so I stacked two bidirectional LSTM layers which is illustrated in Figure 3. Also I used two fully connected linear networks: one

type	sequence length	hidden dim	output shape
bi-LSTM	7	256x2	(7x512)
fc			(7x256)
bi-LSTM	7	256x2	(7x512)
fc			(7x37)
softmax			(7x37)

Table 2: configuration summary of RNN Layers.

for linking between two bidirectional LSTM layers, and the other for an embedding layer to embed output into label-size. Finally a softmax layer is connected at the tail of the network to calcualte CTCLoss. The details of Network configuration are shown in Table 2.

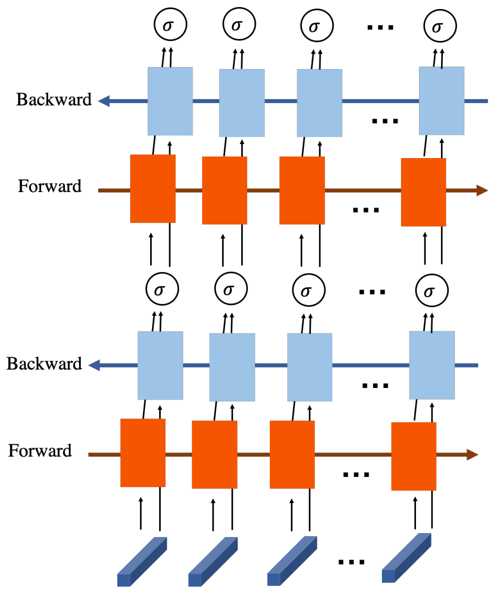

Figure 3: Stacked LSTM layers.

2.3. Measuring Loss

Instead of using cross-entropy erros between one-hot encoded vectors or multi label softmax loss, I adopted the conditional probability method of [3], named CTC(Connectionist Temporal Classification) which is able to train RNNs to label unsegmented sequences directly. This is suitable for the Cap2TxT network since the output of the network doesn't have sequence-wise(time-wise) ground truth labels. The CTCLoss function is already implemented in torch.nn package[12]

3. Experiments & Discussion

3.1. Experiment Details& Results

In all experiment trials, I have never tried any ensemble methods such as taking a average or mean weights of parallel-trained networks. Also, I didn't make any kinds of data augmentation. Of course, I know that the method as ensemble or the method of passing, averaging the same network in parallel and increase training set size by data augmentation greatly improves the performance. I did not apply such methods to this project because those methods show exaggerated accuracy and they are not suitable for evaluating the performance of the network itself.

I trained Cap2TxT with one GPU(RTX 2080 SUPER), and it takes 50 minutes for 300 epochs. I used batch size 32, RMSprop for optimizer as learning rate 0.0001, as mentioned before, and as shown in Table 1, Table 2, the hidden sequence length in network is 7, the hidden variable dimension of LSTM is 256(actually 256*2 since it uses bidirectional LSTM). The results are shown in Table 4.(The bold letter is my final model.)

During the experiments, I tracked the failed case. The most unsuccessful case was confusing '5' and 's'. For example, the CAPTCHA image in Figure 1, even I failed to answer correct. Next most failed cases are confusing '2' and 'z', confusing 'v' and 'y' and confusing '0' and 'o'. Thus, I expect that replacing CNN model to more powerful model one(like ResNet) can make it better.

3.2. Trials not Adopted as Final Model

I tried some other methods before adopting the final Cap2TxT Network.

Candidate 1 was the first trial of this project. It used CNN model shown in Table 3. I tried to make model as light as possible. Then I connected the output from CNN layers into one uni-directional LSTM layer. Finally, I used cross entropy loss. This model didn't work well. I thought choosing wrong loss function was critical.

Candidate 2 is the first trial to use CTC loss. It also used the CNN layers that have configurations shown in Table 3. Then I connected the output to one- layer unidirectional LSTM. The performance was significantly increased than Candidate 1.

Candidate 3 has same CNN layers with Candidate 1 and 2. Then I connected the output to two layered bi-directional LSTM(The same layers with final model).

Candidate 4 is same with Cap2TxT except the sequence length. The CNN layer outputs of Candidate 4's length is 10. I thought the longer sequence length would work better, but it was not. Increased parameters to learn may be the cause of bad performance.

Candidate 5 - 8 have exactly same model with Cap2TxT. But Candidate 5 used learning rate 0.0004, Candidate 6

Type	Configs
Convolution	k:3,s:1,p:1
MaxPool	k:2
Convolution	k:3,s:1,p:1
MaxPool	k:2
Convolution	k:3,s:1,p:1
BatchNorm	
Convolution	k:3,s:1,p:1
MaxPool	k:2,s:(2,1),p:(0,1)
Convolution	k:3,s:1,p:1
BatchNorm	
Convolution	k:3,s:1,p:1
MaxPool	k:2,s:(2,1),p:(0,1)
Convolution	k:2,s:1,p:0
BatchNorm	
AvgPool	k:3,s:1

Table 3: Trial CNN model. k : Kernel size, s : stride, p : padding,

used learning rate 0.001, Candidate7 used batch size 64, and Candidate7 used adam optimizer(lr=0.0001, beta=0.5).

The summary of experiment results for aforementioned models are in Table 4.

Networks	char accuracy	word accuracy
Candidate1	37.1%	0.0%
Candidate2	66.3%	46.2%
Candidate3	77.6%	62.6%
Candidate4	85.9%	72.6%
Candidate5	84.5%	72.6%
Candidate6	84.8%	72.8%
Candidate7	86.6%	76.1%
Candidate8	87.8%	77.1%
Cap2TxT	**88.2%**	**77.9%**

Table 4: Recognition accuracy(%) on test dataset with final Cap2TxT and other candidate network models.

4. Conclusion

In this paper, I have proposed an end-to-end deep neural network framework, called *Cap2TxT*, which interprets the CAPTCHA image and returns the text sequence in it. Cap2TxT consists of a stacked layers of deep CNN, RNN layers. Cap2TxT can take any size of images, and it is able to produce any length of sequence. The experiment results for the network says that this model shows good performance(char accuracy 88.2%, word accuracy 77.9%) for random-wise CAPTCHA text sequence transcription. Furthermore, by modifying the alphabet and output size of CNN layers, this model can be applied into any character, any length of text sequence recognition. Even if it would recognize any type of sequence information in image.(*e.g.* musical notes) But this network can takes input only for images standing upright. It cannot handle the images that are upside down or mirrored. Thus, developing the network that can handle distorted, mirrored and flipped images and developing more robust and light weight fast inferring network are left as future works.

Acknowledgement

Everything I've learned in this class has helped me figure out what I can see myself doing as a career because I genuinely enjoy what I learn from your class. Thank you so much for everything during the class.

References

[1] S. M. R. R. Bhavani. A survey on coding algorithms in medical image compression. 2010.

[2] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints*, page arXiv:1406.1078, June 2014.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[6] N. Islam, Z. Islam, and N. Noor. A Survey on Optical Character Recognition System. *arXiv e-prints*, page arXiv:1710.05703, Oct. 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Q. V. Le, N. Jaitly, and G. E. Hinton. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv e-prints*, page arXiv:1504.00941, Apr. 2015.

[9] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In J. Mundy, R. Cipolla, D. Forsyth, and V. di Gesu, editors, *Shape, Contour and Grouping in Computer Vision*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 319–345. Springer Verlag, Jan. 1999. International Workshop on Shape, Contour and Grouping in Computer Vision ; Conference date: 26-05-1998 Through 29-05-1998.

[10] J. Memon, M. Sami, and R. A. Khan. Handwritten Optical Character Recognition (OCR): A Comprehensive Sys-

tematic Literature Review (SLR). *arXiv e-prints*, page arXiv:2001.00139, Dec. 2019.

[11] D. M.Sathya Deepa. Image compression using mh encoding. In *International Journal of Computer Trends and Technology (IJCTT) V13(2):68-71*, July 2014.

[12] PyTorch. https://pytorch.org/docs/master/generated/torch.nn.ctcloss.html.

[13] S. Sathasivam and W. A. T. W. Abdullah. Logic Learning in Hopfield Networks. *arXiv e-prints*, page arXiv:0804.4075, Apr. 2008.

[14] . C. J. Shen, H. Towards a real time system for finding and reading signs for visually impaired users. In *TComputers Helping People with Special Needs.*, 2012.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.