Inpainting Task on Chinese Street View Text

Juan Li
N17186300
jl10889@nyu.edu

Mingxi Chen
N17122277
mc7805@nyu.edu

1. Introduction

Text is one of the easiest ways to save and make use of data. While in the real world, many text data are not directly saved in the form of text, they may be saved as images. If we can extract text information from those images, it will save us a lot of work. Such task is called optical character recognition (OCR), which is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast) [6].

With the help of deep learning and computer vision technology, we can fully make use of OCR. We can apply it to many downstream tasks to help human to do something efficient. A good example of OCR application is scanning receipts. We can take a picture of the receipt, this application can extract key texts from receipts and invoices and save the texts to structured documents and then the structured documents can be used to record our expenditure. As the paper [8], they use a model "Convolutional Universal Text Information Extractor (CUTIE)", which applies convolutional neural networks on gridded texts where texts are embedded as features with semantical connotations, and get a state of the art performance of extract information from receipt images.

We want to do the task of recoginize the Chinese characters on signboard. While in the reality world, the images of signboard may not be so good as the angle or the distance of taking pictures, many factors may result in that the image we need to recognize do not have good quality. Also, unlike English context, if we want to recognizd chinese characters, there are thousands of characters comparing with 26 alphabets. So that if we use the traditional supervised learning method, we may need an extremely large labeled dataset. However, it is so costly to get such labeled dataset and also hard to be scaled up. While unlabelled data is much cheaper to collect, we may also collect tons of data from the Internet for free. In this case, we can take advantage of self-supervised learning. Self-supervised learning empowers us to make use of tons of labelled data that come with the data for free.

According to the method introduced in paper [4], given an image with a missing region, they trained a convolutional neural network to regress to the missing pixel values. They called their model "context encoder". This is a classic application of self-supervised learning. They inpainted the image and fill the hole with specific color, and use the inpainted image and the part been inpainted as data and label pairs just like traditional supervised learning. As a result, their model can generate images conditioned on context advance the state of the art in semantic inpainting, at the same time learn feature representations that are competitive with other models trained with auxiliary supervision.

In our project, we adopt the methodology in [4]. Our main task is to recognize chinese characters in signboard, but we only have a limited amount of labelled data. So we train a pre-trained model using self-supervised learning first. In the procedure of getting pre-trained model, we collect some signboard images without text label, and pick three squares and inpaint them with a specific color. And then, use the inpainted image to generate the original image

2. Related work

There are many research have used the method of self-supervised learning, which has greatly reduced the usage of labelled data. The pre-trained model from self-supervised learning get some improvements in the downstream task. We briefly review the related work in self-supervised learning field.

2.1. Context encoders

They [4] present an unsupervised visual feature learning algorithm driven by context-based pixel prediction. They propose Context Encodes – a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. Their generation model is able to inpaint semantically meaningful content in a parametric fashion, as well as provide a better feature for nearest neighbor-based inpainting methods. They quantitatively demonstrate the effectiveness of our learned features for

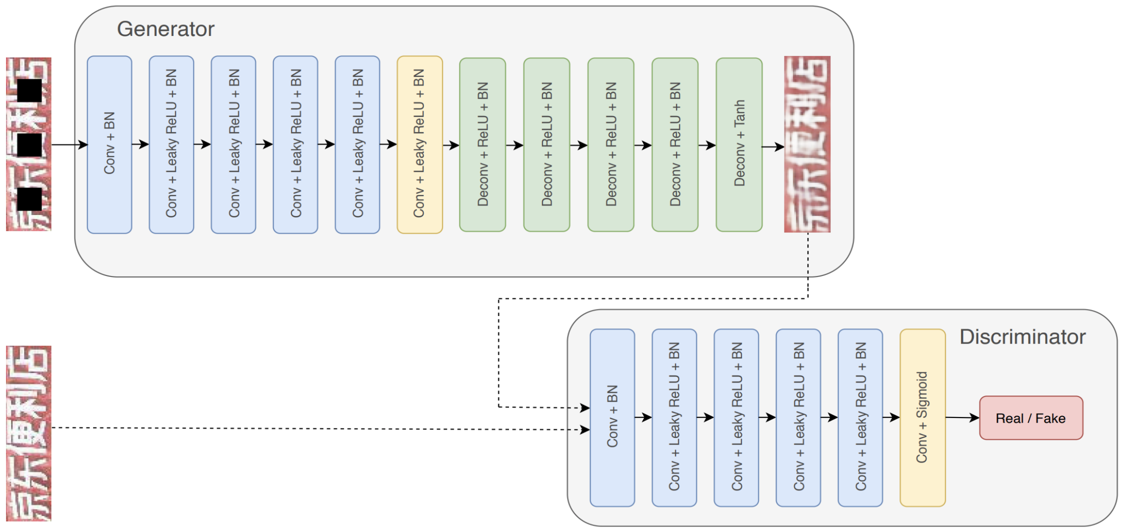


Figure 1: The architecture of non-local attention module and spatial RNN.

CNN pre-training on classification, detection, and segmentation tasks.

2.2. Split-Brain Autoencoders

They [7] propose split-brain autoencoders for unsupervised representation learning. Their method is to split the network into two disjoint sub-networks. Each sub-networks is trained to perform a difficult task - predicting one subset of the data channels from another. Together, the sub-networks extract features from the entire input. By forcing the network to solve cross-channel prediction tasks, they induce a representation within the network which transfers well to other tasks.

3. Methodology

We now give an overview of our architecture and then provide details of mask strategy. As shown in Fig. 1 and Tab 1, the overall architecture is modified from Context Encoder [4] which contains two parts: generative network and discriminative network. The generative network learns to fill in several region dropouts in the input image, while the discriminative network try to distinguish candidates produced by the generator from the true data.

3.1. Generative Network

The goal of the generator is to generate passable refilled images: to lie without being caught. Follow the idea of encoder-decoder, an encoder is a network takes the input and outputs the corresponding feature maps which contain high level information to represent inputs. An decoder is a network takes the feature maps generated by encoder and try to map them to certain outputs. This technique has been widely used in translation, generative models and so on.

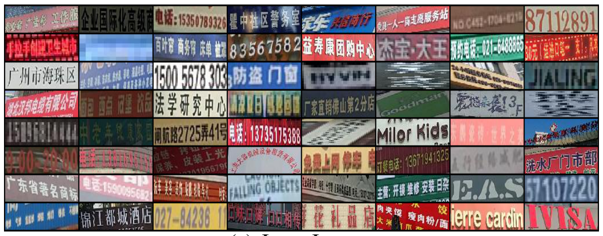
Given an input image of size 32×128, we use the first five stride 2 convolution layers to compute a $512 \times 1 \times 4$ feature representation. Each layer is followed by a Leaky ReLU activation function with batch normalization. Since it's necessary to propagate information from one corner of the feature map to another and fully-connected layer will introduce too many parameters, we apply two convolution layers with 1×1 kernel size to replace fully-connected layer. As for decoder, we use 5 stride 2 transposed convolution layers to do deconvolution operation on the feature representation, each with a ReLU activation function and batch normalization. After a serial of up sampling, the final size of feature map returns to 32×128 which can be regarded as the corresponding ”fake” image generate by our generative network.

3.2. Discriminative Network

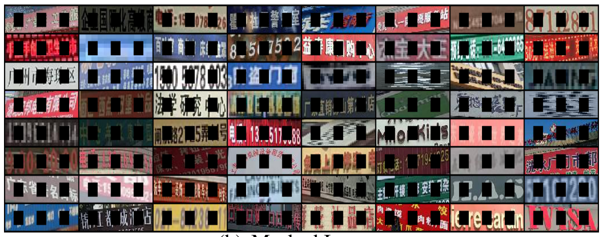
The goal of the discriminator is to identify image generated by generator as fake. Similar to encoder, after five convolution layers with Leaky ReLU and batch normalization, the network generate $512 \times 1 \times 4$ feature representation, then we apply one 1×4 convolution layer followed by Sigmoid to compute Boolean label, 0 represents fake and 1 represents true.

3.3. Mask Strategy

In contrast to Context Encoder [4], we choose to drop out 3 square patches in the image and the size of those regions



(a) Input Images



(b) Masked Images

Figure 2: Examples of our mask strategy.

is: 16×16. The coordinate of each region's left upper corner (from left to right) is: (8, 15), (8,55) and (8, 95). As shown in Fig. 2, this mask strategy fits well to images contained text line. The mask regions hide both background and character information and force our model learning from context and filling with appropriate pixels. Moreover, mask three regions at the same time will encourage our model capturing long range dependencies, which is important to downstream tasks such as optical character recognition.

3.4. Loss Function

Since the goal of generative network is to confuse discriminative network by producing samples as real as input images, our loss function contains two parts: one is reconstruction loss which is responsible for capturing the structure of the missing parts and handle continuity within context. We use a normalized masked L2 distance:

$$\mathcal{L}_{\text{rec}} = \| M \odot (x - G((1 - M) \odot x)) \|_2^2 \tag{1}$$

However, L2 distance often leads to blurry results. Therefore, we combine it with adversarial loss:

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{x \in \mathcal{X}}[\log D(x) + \log(1 - D(G((1-M) \odot x))] \tag{2}$$

where $\odot$ is element-wise product operation, D represents the discriminative network, G represents the generative network and M is the mask. Then the joint loss is:

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{\text{rec}} + \lambda_{adv}\mathcal{L}_{\text{adv}} \tag{3}$$

Since the main objective of the generator is to reconstruct the masked image, therefore the network should focus more on the reconstruction loss and let the adversarial loss to assist the generator avoiding blurring images.

Layer Name	Output Size	Details
Cnn_1	$64 \times 16 \times 64$	4×4 64
Cnn_2	$64 \times 8 \times 32$	4×4 64
Cnn_3	$128 \times 4 \times 16$	4×4 128
Cnn_4	$256 \times 2 \times 8$	4×4 256
Cnn_5	$512 \times 1 \times 4$	4×4 512
Cnn_6	$100 \times 1 \times 4$	1×1 100
Deconv_1	$512 \times 1 \times 4$	1×1 512
Deconv_2	$256 \times 2 \times 8$	4×4 256
Deconv_3	$128 \times 4 \times 16$	4×4 128
Deconv_4	$64 \times 8 \times 32$	4×4 64
Deconv_5	$64 \times 16 \times 64$	4×4 64
Deconv_6	$3 \times 32 \times 128$	4×4 3

Table 1: Architecture of our proposed generative network

4. Experiment Results

4.1. Impletmentation Details

4.1.1 Pre-trained Model

Dataset. Dataset used for pre-trained comes from the ICDAR2019 [2]. We used the full annotated images from the whole dataset, which have crop the images only contain texts. There are 10,000 images in total, all of them are rectangle but in different sizes. The following are some samples of our pre-trained data.

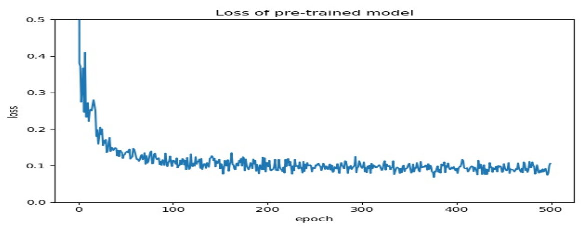
Inpainting and hole-filling . We cropped three squares of 16*16 in the same position of each image, and filled the holes with specific color.

Image generation. We use the Context Encoders [4] pipeline. Our pipeline was implemented in Pytorch. We resize all images to 32*128. The batch size is 64, learning rate is 0.0002 and we choose ADAM [3] as our optimization method. As our loss function is composed of two elements, one is reconstruction loss, and the other is adversarial loss. The weights of reconstruction loss and adversarial loss are 0.998 and 0.992. So our loss function can be represented as $L = 0.998L_{rec} + 0.002L_{adv}$.

4.1.2 Downstream Task Model

Dataset. The dataset for training a model to do the image-based sequence recognition task comes from the competition of ICDAR2019 Robust Reading Challenge on Reading Chinese Text on Signboard [1]. We have around 50,000 images with corresponding texts labels. We split the total dataset as 95% training data and 5% test data.

Model implementation. We mainly use the image-based sequence recognition model – CRNN for the training of our downstream task. We use Pytorch to implement the pipeline [5]. We resize all images to 32*128.



(a) Loss of pre-trained model

Figure 3: Loss of pre-trained model.

4.2. Results

4.2.1 Pre-trained Model

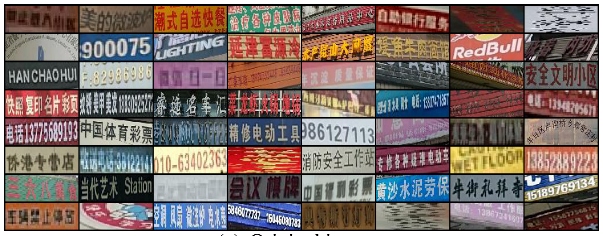
In our pre-trained model, our task is to generate images to fill the regions we inpainted. So the evaluation of the generating model is compare the original image and the image generated by the pre-trained model. The loss is the difference between them. Figure 3 is the loss during the training. We can clearly see that the loss is decreasing and going to converge. Fig. 4 shows some results generated by our generator. As can be seen in Fig. 4, our model does quiet well on both numbers, English and Chinese characters with a variety of fonts, backgrounds and perspective situations. It demonstrates that our model has learned some implicit knowledge of different structures, spatial relationships and dependencies within a single character or between two characters.

4.2.2 Recognition Task

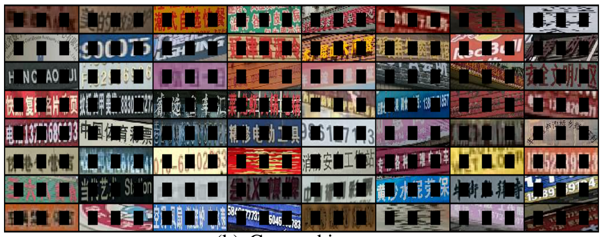
Our main task is to recognize Chinese characters sequence on signboard images. Our pre-trained model should work with a specific model to do our downstream task, we chose CRNN [5], a combination of CNN, RNN and CTC loss, and it has a good performance in image-based sequence recognition tasks. We concatenate our pre-trained model and the CRNN model to train on our recognition task. When initialize the model parameters, we import all parameters of pre-trained model we trained before, and for the CRNN part, randomly initialize it's parameters.

During training, the parameters of pre-trained part model would be fine tuned, while the CRNN part model parameters will be trained from the very beginning. In order to evaluate the performance of pre-trained model, we also train on the pure CRNN model(do not concatenated with the pre-trained model).

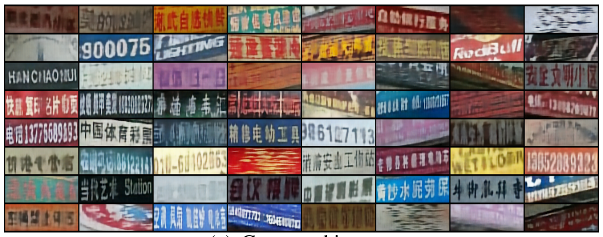
For both models, our pipeline was implemented in Pytorch. We resize all images to 32*128. The batch size is 64, learning rate is 0.0002 and we choose ADAM [3] as our optimization method. As our loss function is composed of two elements, one is reconstruction loss, and the other is adversarial loss. The weights of reconstruction loss and



(a) Original images



(b) Cropped images



(c) Generated images

Figure 4: Images generated from pre-trained model.

Method	Supervision	Accuracy
CRNN	Initialization	0.752734
Ours	Context	**0.8016**

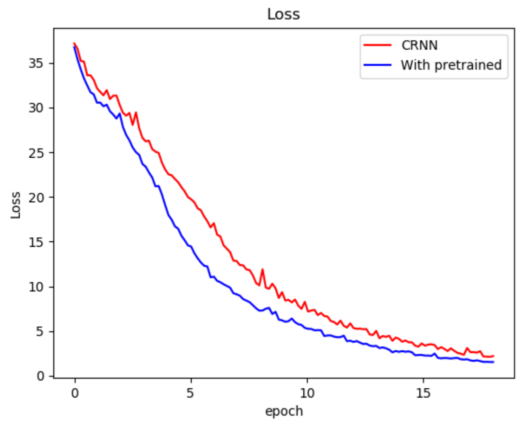
Table 2: Quantity comparison of character recognition task.

adversarial loss are 0.998 and 0.992. So our loss function can be represented as $L = 0.998L_{rec} + 0.002L_{adv}$.

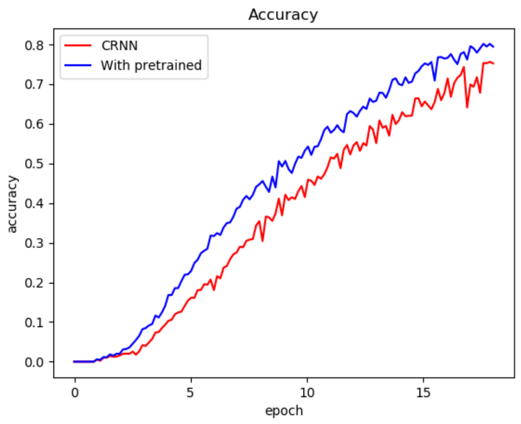
After training for around 20 hours, we got the results shown in Fig. 5 and Tab. 2. At last, our accuracy of mode with pre-trained part is 0.8016, which is almost 0.05 better than the pure CRNN model. The result shows that our pre-trained model helps a lot in the Chinese characters sequence recognition task.

5. Conclusion

Our idea is to do the inpainting task on Chinese Characters Dataset. After modification on Context Encoder [4], our model achieves quite well result which successfully refilled masked input. The performance is stable on different situations. We also apply OCR as downstream task to evaluate features learned by this unsupervised task. The accuracy of our model preforms roughly 5% above random initialized CRNN model which further demonstrated that our model



(a) Input Images



(b) Masked Images

Figure 5: Comparison of loss and accuracy .

have learned implicit knowledge of different structures and spatial relationships.

References

[1] Icdar 2019 robust reading challenge on reading chinese text on signboard, 2019. [Online; accessed 14-December-2019]. 3

[2] Icdar2019 robust reading challenge on large-scale street view text with partial labeling, 2019. [Online; accessed 14-December-2019]. 3

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3, 4

[4] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 1, 2, 3, 4

[5] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(11):2298–2304, 2017. 3, 4

[6] Wikipedia contributors. Optical character recognition — Wikipedia, the free encyclopedia, 2019. [Online; accessed 14-December-2019]. 1

[7] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017. 2

[8] X. Zhao, Z. Wu, and X. Wang. Cutie: Learning to understand documents with convolutional universal text information extractor. *arXiv preprint arXiv:1903.12363*, 2019. 1