Resolution Enhancement for Low-resolution Text Images Using Generative Adversarial Network

Jie Kong^a and Congying Wang

Xi'an Shiyou University, School of Computer Science, No. 18 2nd Dianzi Road, Xian, China

Abstract. In recent years, although Optical Character Recognition (OCR) has made considerable progress, low-resolution text images commonly appearing in many scenarios may still cause errors in recognition. For this problem, the technique of Generative Adversarial Network in super-resolution processing is applied to enhance the resolution of low-quality text images in this study. The principle and the implementation in TensorFlow of this technique are introduced. On this basis, a system is proposed to perform the resolution enhancement and OCR for low-resolution text images. The experimental results indicate that this technique could significantly improve the accuracy, reduce the error rate and false rejection rate of low-resolution text images identification.

1 Introduction

In recent years, OCR has been widely applied in the information input of data records on the printed paper. OCR is the process of converting text image, such as the text on the handwriting document, printed document, scanned document, etc., to the machine-encoded text [1]. It enables the above-mentioned text to be edited, searched, stored digitally, displayed online and used in machine processing, such as cognitive computing, machine translation, text to speech and text mining.

However, some OCR recognition systems may produce errors in recognizing low-resolution text images. This is because low-resolution text images lack high-frequency image details, which makes it difficult for OCR systems to retrieve text information correctly. This problem exists widely in practical applications. For instance, text images generated many years ago may be limited by sampling devices and encoding algorithms, resulting in low-resolution, text in photos and videos may also result in low-resolution after clipping and enlargement, which make the traditional OCR recognition technology unable to fulfil the corresponding requirements.

A solution for this problem is to perform super-resolution processing on low-resolution text images, so as to achieve accurate recognition for the OCR [2]. As a classical topic in computer image processing, super-resolution processing is a general term for techniques which could enhance the resolution of images [3]. With the rapid development of machine learning and pattern recognition techniques in recent years, lots of related techniques have been applied in the field of super-resolution processing, and achieved good recognition performance for the OCR. In this study, an emerging machine learning technique: generative adversarial

network (GAN) is adopted to build a super-resolution processing system to improve the performance of OCR recognition.

The contribution of this study includes the following contents. The architecture of generative adversarial networks based super-resolution processing system, as well as the loss function for this system is implemented by TensorFlow. The performance of proposed system is evaluated by test datasets.

2 Related works

As early as the late 20th Century, researchers began to pay attention to the problem of low-resolution text image recognition. Some traditional approaches have been adopted to recognize the degraded text, for instance, the deformation of elastic templates [4] and n-grams [5]. In recent years, the super-resolution processing for OCR of low-resolution images has attracted wide attention from academia, as a result, more solutions have been proposed, such as the multi-scale binarization framework [6], the Anchored Neighbourhood Regression (ANR) [7] and the Simple Functions (SF) [8]. Especially with the rising of machine learning techniques, lots of convolutional neural network (CNN) related studies have been proposed to solve the problem and have achieved remarkable performance [9]. Residue learning and adaptive gradient clipping is applied by Kim et al. to build a 20 layers CNN to do the super-resolution processing, which shows the best performance at that time [10]. Zhang et al. use the gradient descent based weighted-mean-squared-error loss function on the CNN for super-resolution reconstruction [11]. This work has achieved an OCR accurate of 78.10% on the ICDAR2015-TEXTSR dataset, which is very close to the OCR accurate of original high-resolution images (78.80%).

^a Corresponding author: jkong@xsyu.edu.cn

3 The GAN based super-resolution processing

In this study, an emerging machine learning technique GAN is adopted perform super-resolution processing, so as to find a new solution for OCR of low-resolution text images.

3.1 The mathematical model of GAN

GAN is an unsupervised learning training algorithm proposed by Ian Goodfellow et al. in 2014 [12]. There are two neural networks in the model of GAN, one is named Generator (G), and the other is named Discriminator (D). The main idea is inspired by the zero-sum game between the two networks, so as to achieve the best generating performance.

At first, there is a 1st version of Neural Network Generator (NNGenerator V1), which generates poor quality images, and then there is a 1st version of Discriminator network (Discriminator V1), which can accurately classify the generated pictures and the real pictures. In short, the Discriminator is a binary classifier, which outputs 0 for images generated by neural network and 1 for real inputted images. Next, a 2nd version of Neural Network Generator (NN Generator V2) is trained to produce a slightly better image, allowing the Discriminator V1 to think that the generated images are real, and then a 2nd version of Discriminator network (Discriminator V2) is trained, which could accurately classify the real images and images generated by NN Generator V2. By iterating the process above, there will be the 3rd, 4th...nth version of Neural Network Generators and Discriminator networks. In the end, the Discriminator network is unable to classify the generated pictures from the real pictures, thus the network is fitted.

The mathematical representation of the GAN is shown below.

$$V(D, G) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(G(x)))] \quad (1)$$

$V(D, G)$ is the value function of zero-sum game between D and G . P_{data} refers to the distribution of real image sets, and P_G is the distribution of images generated by G . The objective of GAN is to find the optimal solution for minimize the difference between P_{data} and P_G , which is defined as:

$$G^* = \arg \min_G \max_D V(G, D). \quad (2)$$

3.2 The implementation of Generator

According to the SRGAN algorithm proposed by Ledig et al. [13], the Generator G is implemented through the TensorFlow in this study. The architecture of Generator is shown in figure 1.

Figure 1. The architecture of Generator G

3.2.1 The input layer

The function of the input layer in the Generator is to pre-process the image data, including reading the image file, decoding, regularization and cropping. The process of input layer in TensorFlow is demonstrated in table 1.

Table 1. The process of input layer

Step	Function	Comment
1	convert_to_tensor()	Convert Python object to TensorFlow available tensor
2	decode_png()	Decode PNG images
3	Convert_image_dtype()	Normalize the data in tensors to floating number data between 0 and 1
4	Crop_to_bounding_box()	Crop the images for training

3.2.2 The convolution layers

In the implementation of convolution layers, there are 15 layers of 4 types in the Generator: k9n64s1, k3n64s1, k3n256s1, k9n3s1. In these types, **k** represents the size of the convolution kernel, **n** represents the number of generated feature graphs, that is, the number of convolution kernels, **s** represents the step size of the convolution kernel.

3.2.3 The activation function

8 Parametric Rectified Linear Unit (PReLU) activation functions are used in the generator to adaptively learn parameters from the data. PReLU has the characteristics of fast convergence speed and low error rate. It can be used for training of backpropagation and optimization with other layers. The PReLU activation function is denoted as:

$$f(y_i) = \begin{cases} y_i, & \text{if } (y_i > 0) \\ a_i y_i, & \text{if } (y_i \leq 0) \end{cases} \quad (3)$$

Since there is no PReLU activation function interface available in TensorFlow, improvements have been made on the ReLU interface. Two parameters named *pos* and *neg* are set in the program. In the program, *alphas* is the a_i in equation (3). The output of the activation function is the value of *pos* + *neg*.

```

pos = tf.nn.relu(inputs)
neg = alphas * (inputs - abs(inputs)) * 0.5
  
```

3.2.4 The B Residual Block

Each B residual block contains two 3*3 convolution layers, and the convolution layer is followed by the batch normalization layer and PReLU used as activation functions, as well as two sub-pixel convolution layers for increasing the feature size.

In addition to the convolution layer and PReLU function described above, the Batch Normalization layer is responsible for resolving changes in the distribution of data in the middle layer during training. During the training process, the whole distribution of the deep neural network gradually approaches the upper and lower bounds of the value range of the nonlinear function, resulting in the disappearance of the gradient of the low-level neural network when it propagates backward. This is the essential reason for the slow convergence of the training deep neural network. For each hidden layer neuron, Batch Normalization draws the input distribution which is gradually approaching the limit saturation region of the value range after mapping the nonlinear function back to the normal distribution with the mean value of 0 and the variance value of 1, so that the input value of the nonlinear transformation function falls into the input sensitive region, avoiding the gradient disappearance problem. To achieve the Batch Normalization algorithm, `tf.nn.batch_normalization()` method in TensorFlow is adopted in this study.

The sub-pixel convolution layer is responsible for extracting multi-channel features from low-quality images, and then merging these features into super-resolution images by some means. In this study, the algorithm of the sub-pixel convolution layers is shown in equation (4), where I^{SR} and I^{LR} denote the super-resolution images and low-resolution images, W_L and b_L are learnable network weights and biases of layer L, f^L is the function of layer L.

$$I^{SR} = f^L(I^{LR}) = PS(W_L * f^{L-1}(I^{LR}) + b_L) \quad (4)$$

According to this equation, the super-resolution images is obtained through the PS (Periodic Shuffling) operator, which periodically inserts the low resolution features into high resolution images according to specific locations.

To implement the sub-pixel convolution layer in TensorFlow, the function `tf.transpose()` method is used firstly to transpose the tensor on each channel, then the `tf.reshape()` method is used to convert the size of the image, and finally `tf.concat()` is used to merge the tensor on each channel, which completes the Periodic Shuffling operation and outputs the desired size of the image tensor.

3.3 The implementation of Discriminator

The architecture of Discriminator is shown in Figure 2, which consists of convolution layer, the LeakyReLU activation function, the dense layer, and the sigmoid activation function. Some layers have been described in detail in the previous section, therefore, only the design and implementation of the two activation functions and the dense layer will be introduced in this section.

Figure 2. The architecture of Discriminator D

3.3.1 The activation function

There are two kinds of activation functions applied in Discriminator: the LeakyReLU and the Sigmoid. The LeakyReLU activation function is a maximum value function, as shown in equation (5).

$$f(y_i) = \begin{cases} y_i, & \text{if } (y_i > 0) \\ 0.01y_i, & \text{if } (y_i \leq 0) \end{cases} \quad (5)$$

Leaky ReLU solves the Dead LeLU problem, in which certain neurons may never be activated, resulting in the corresponding parameters never being updated. In addition, the convergence speed of LeakyReLU is much faster than that of other activation functions such as sigmoid. In this study, the LeakyReLU activation function is implemented by the `Keras.layers.LeakyReLU().Call()` method in TensorFlow.

Sigmoid is a widely used activation function in neural network due to the monotone increasing nature of it and its inverse function. It could map the input of continuous real values between 0 and 1. The definition of the Sigmoid is shown in equation (6). In this study, it is implemented by the `Tf.nn.sigmoid()` method in TensorFlow.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

3.3.2 The dense layer

The dense layer is responsible for connecting all the nodes in its input layer and output layer. For instance, the hidden layer in figure 3 is a dense layer.

Figure 3. Schematic diagram of dense layer

In the design of Discriminator, 2 dense layer are applied, one using 1024 neuron nodes, and the other using only 1 neuron node, which is activated through the Sigmoid to discriminate the real images.

4 the implementation of loss function

The loss function is used to the cost of an event through mapping the variable(s) of this event to a real number. It is widely applied in optimization, decision theory,

machine learning, et al. A better learning model can be obtained by training the loss function. In other words, the loss function could measure the performance of the learning model, which makes the loss function play an important role in machine learning.

The loss function applied in this study is shown in equation 7, which consists of two parts: the content loss and the adversarial loss denoted as l_{Con}^{SR} and l_{Adv}^{SR} .

$$l^{SR} = l_{Con}^{SR} + 10^{-3} l_{Adv}^{SR} \quad (7)$$

4.1 The content loss function

The content loss function is based on the mean square error loss, and the latter is defined as follows:

$$l_{MSE}^{SR} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2 \quad (8)$$

Equation (8) is the most widely used loss function in image super-resolution. The specific introduction could find in [14]. However, this loss function may lead to the lack of high-frequency contents in images.

For this problem, VGG19 CNN proposed by Simonyan & Zisserman is applied [15]. The trained VGG19 network is used to perform feature extraction. Put the generated image and the corresponding nature high-resolution image into the VGG19 network with the trained weights, take the convolution layer of the last layer and take out the features to do the mean square error, then some high-frequency details can be smoothed. This is the content loss mentioned in this section, which is defined as:

$$l_{MSE}^{SR} = l_{VGG/(5,4)}^{SR} = \frac{1}{W_{5,4} H_{5,4}} \sum_{x=1}^{W_{5,4}} \sum_{y=1}^{H_{5,4}} (\phi_{5,4}(I^{HR})_{5,4} - \phi_{5,4} G_{\theta_G}(I^{LR})_{5,4})^2 \quad (9)$$

in which $\phi_{5,4}$ refers to put image I into VGG19 network and then extract the pixel features of the 4th convolution layer in the 5th sets of the VGG19. This equation calculates the Euclidean distance between low quality image I^{LR} and high quality image I^{HR} , which is implemented in Tensorflow by mathematic methods such as `tf.reduce_mean()`, `tf.reduce_sum()` and `tf.square()`.

4.2 The adversarial loss function

The adversarial loss function is used to make the Generator generates the images that can deceive the Discriminator. It is defined as:

$$l_{Adv}^{SR} = \sum_{n=1}^N -\log D_{\theta_D}(G_{\theta_G}(I^{LR})) \quad (10)$$

In this equation, the $G_{\theta_G}(I^{LR})$ is the images generated by the Generator G from the low resolution images I^{LR} . $D_{\theta_D}(G_{\theta_G}(I^{LR}))$ refers to the probability that the Discriminator D correctly classifies the

$G_{\theta_G}(I^{LR})$. This loss function is complemented by the methods `tf.log()` and `tf.reduce_mean()` in TensorFlow.

4.3 The discriminator loss function

The discriminator loss function is designed based on the equation (1). The objective of this function is to maximize the probability difference between the Discriminator D correctly classifies the natural high-resolution images I^{HR} and the images generated by the Generator G from the low resolution images I^{LR} . The definition of the discriminator loss function is shown in equation (11). The implementation of this function in Tensorflow is also by the built-in mathematical methods such as `tf.log()` and `tf.reduce_mean()`.

$$l_{Dis}^{SR} = -\sum_{n=1}^N [\log D_{\theta_D}(I^{HR}) + \log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))] \quad (11)$$

5 Experiment

5.1 The architecture of resolution enhancement and OCR system for text image

Based on the concepts introduced in section 3 and 4, a system is designed and implemented to perform the resolution enhancement and OCR for low-resolution text images. The architecture of this system is shown in Figure 4.

Figure 4. The architecture of proposed system

The experiment is performed on the RAISE [16] dataset, which is a real world image data set and mainly used to evaluate digital forgery detection algorithm. The SRGAN model obtained from the training module is used to enhance the resolution of inputted text images. After the generation of super-resolution text images, the OCR module implemented by Tesseract-OCR engine is used to recognize the super-resolution text images.

5.2 The training of GAN

Because of the complexity of GAN, in order to improve the performance of the training module, it is necessary to initialize the GAN by the pre-trained parameters, and adjust the training parameters of the network in the training process. The parameters setting applied in the training is shown in table 2.

Table 2. The parameters setting applied in the training

Parameters	Value	Comment
display_freq	20	frequency of writes
pre_trained_model	True	Is there a pre trained model?
vgg_scaling	0.006100	Generalization value in loss function
save_freq	10000	Preservation frequency of model weight
ratio	0.001000	The weight of discriminator loss
queue_thread	10	Number of threads
decay_rate	0.100000	The decay rate of learning
decay_step	100000	The steps of decay
learning_rate	0.000100	The rate of learning
max_iter	200000	Iterations of training
batch_size	16	The size of batch
summary_freq	100	The frequency of writing summary

The variation diagram of the content loss function, the adversarial loss function and the discriminator loss function is demonstrated in figure 5 to 7.

content_loss

Figure 5. The variation diagram of content loss function

In figure 5, the content loss function is iterated 200000 times. After the iteration starts, the loss decreases dramatically, which is the advantage of using a pre-processed model. When the iteration reaches about 120000 steps, the loss value begins to balance gradually until the end of iteration.

In figure 6, with the continuous rise of adversarial loss, more images generated by Generator can deceive the Discriminator. After the 110000 steps, the adversarial loss is gradually stable.

adversarial_loss

Figure 6. The variation diagram of adversarial loss function

discriminator_loss_1

Figure 7. The variation diagram of discriminator loss function

In figure 7, the value of discriminator loss is getting lower from the start to the 100000 steps, which means that the probability of the Discriminator correctly classify the images is becoming lower. Finally, it tends to balance near the loss value of 0.2 until the end of training, which means the Discriminator has considered the generated image to be the original high quality image.

5.3 The performance of resolution enhancement and OCR for text image

To validate the performance of the resolution enhancement and OCR system for text image, 60 text images are selected as the test dataset, which consists of 20 original high-resolution text images, 20 low-resolution text images, and 20 super-resolution text images reconstructed from low-resolution text images by the resolution enhancement module. Examples of some text images in the test dataset are shown in figure 8.

In this study, the performance of the resolution enhancement is evaluated by the average accuracy, average error rate and average false rejection rate of the above-mentioned images by OCR. The experimental results are shown in table 3, where the LR, SR and HR is the abbreviation Low-Resolution, Super-Resolution and High-Resolution.

conventional weapons are difficult to use

They can only strike a limited number

(a) Original high-resolution text image

conventional weapons are difficult to use

They can only strike a limited number

(b) Low-resolution text image

conventional weapons are difficult to use
They can only strike a limited number

(c) Super-resolution text image

Figure 8. Examples of some text images in the test dataset

The experiment results indicate that the super-resolution processing for the low-quality text images has improve the average accuracy of OCR by 61.57%, and it is close to the average accuracy of the original high-resolution images (96.03% to 98.90%). The average error rate is also reduced from the 47.70% of low-resolution text images to 6.85% of super-resolution text images. But the average error rate of super-resolution text images is still far from the original ones (0.068%). The average false rejection rate reduces from 18.70% of low-resolution text images to -2.94% of super-resolution text images. The reason for the negative rejection rate of SR is that OCR divides one word in for several times, which leads to the increase of recognition characters. In summary, super-resolution processing enhances the OCR performance of low-quality text image significantly.

Table 3. The experimental results

Average number of correctly recognized characters			Average accuracy		
LR	SR	HR	LR	SR	HR
56.9	163.2	166.5	0.3446	0.9603	0.9890
Average number of falsely recognized characters			Average error rate		
LR	SR	HR	LR	SR	HR
80.6	11.7	0.9	0.4770	0.0685	0.0068
Average number of falsely rejected characters			Average false rejection rate		
LR	SR	HR	LR	SR	HR
31.8	-5.1	0.7	0.1870	-0.0294	0.004

6 Conclusion

The recognition for low-resolution text images has strong theoretical significance and application value. In this study, the GAN based super-resolution processing is applied to enhance the OCR performance of low-quality text images. The experiment results indicate that this technique could significantly improve the accuracy, reduce the error rate and false rejection rate of text images identification.

Acknowledgement

This work is supported by the Science Research Project of Shaanxi Provincial Department of Education (Grant No: 17JK0614) and the Youth Innovation Fund of Xian Shiyou University (Grant No: 2013BS025).

References

1. A. Marial & J. Jibrael, *Int. J. Appl. Eng. Res*, **12**, 1129 (2017)
2. A. Abedi & E. Kabir, *Multimedia Tools and Applications*, **76**, 16415 (2017)
3. K. Nasrollahi & T. B. Moeslund, *Machine vision and applications*, **25**, 1423 (2014)
4. A. Del Bimbo, S. Santini, & J. Sanz, *In Proceedings of 12th International Conference on Pattern Recognition*, 433 (1994)
5. S. M. Harding, W. B. Croft, & C. Weir, *In International Conference on Theory and Practice of Digital Libraries*, 345 (1997)
6. R. F. Moghaddam & M. Cheriet, *Pattern Recognition*, **43**, 2186 (2010)
7. J. Jiang, X. Ma, C. Chen, T. Lu, Z. Wang, & J. Ma, *IEEE Transactions on Multimedia*, **19**, 15 (2016)
8. C. Y. Yang & M. H. Yang, *In Proceedings of the IEEE international conference on computer vision*, 561 (2013)
9. C. Dong, X. Zhu, Y. Deng, C. C. Loy, & Y. Qiao, *arXiv preprint, arXiv:1506.02211*, (2015)
10. J. Kim, J. Kwon Lee, & K. Mu Lee, *CVPR*, 1646 (2016)
11. H. Zhang, D. Liu, & Z. Xiong, *VCIP*, 1. (2017)
12. I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., *NIPS*, 2672 (2014)
13. C. Ledig, Z. Wang, W. Shi, et al., *CVPR*, 105 (2017).
14. W. Shi, J. Caballero, F. Huszar, et al., *CVPR*, 1874 (2016)
15. K. Simonyan & A. Zisserman. *arXiv preprint, arXiv:1409.1556*, (2014)
16. D. T. Dang-Nguyen, C. Pasquini, V. Conotter, et al., *In Proceedings of the 6th ACM Multimedia Systems Conference*, 219 (2015)