In-game architectural image translation using improved Cycle-Gan

Bochi Meng [1]

[1]School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, UK

bochimeng@gmail.com

Abstract. The point of video games is that players can reap success and excitement in games that they cannot easily experience in the real world. The quality of translating in-game architectural images to photos determines whether the game has many players and good prospects for development. This research in this paper is to implement the function of image-to-image translation using Cycle-GAN. In this work, the dataset is pre-processed to make it more suitable for training the network. Then images are generated by the generator, and the discriminator determines whether the generated ones seem real or not. The confrontation loss and the cycle loss are performed to constrain the learning of the entire system. However, distractions still exist in this system, such as people in the background of the game could be wrongly identified as part of the building, or as a pillar and hence resulting in odd results. To mitigate it, a self-attention mechanism was added to the network to address this phenomenon, allowing the network to focus on the architecture and not disperse attention to some of the game characters. After optimization and testing, the results show that the network can be well-optimized for game-style images to resemble the realistic architecture more closely.

Keywords: video games, Cycle-Gan, optimization, self-attention.

1. Introduction

In today's fast-changing technology, the gaming industry has entered an era of competition. Video games are the result of technological development, which is a development trend [1]. When physical objects cannot exist, virtual ones can meet the needs of people's hearts; when virtual ones are close enough to reality, we can reap success and excitement in games that cannot be easily experienced in the real world [2]. And the optimization of the pictures in the game determines whether the game has a large number of loyal players and good prospects for development [3].

Video games are a form of interactive entertainment. The gamer inputs information to the carrier (console, PC, handheld, mobile phone, etc.), the carrier gives feedback based on the player's input, which is fed back to the player through the output channel (screen), and in the process, the player gets interactive entertainment [4]. The output part is therefore very important and is the most unique and enjoyable part of video games compared to traditional games. If you kick a ball on the field, the feedback you get from the ball is relatively simple and can only form a take-off and drop based on physical rules; it cannot turn into a ball of fire in the air when you kick it, nor does it create a virtual environment specifically for you to interact with strong feedback when you shoot successfully. Video games, on the other hand, are different in that they disseminate a lot of interactive information through the vehicle that outputs it. For example, it lets you play as a famous footballer, allowing you to score a goal and reap the cheers of the crowd, the screams of the hosts and be the saviour of your team [5]. This information is heavily stimulating to your brain and gives you satisfaction. That's the beauty of video games, strong feedback of information. This is the underlying motivation for people to play games, to seek that sense of immersion, to seek that superb feedback that transcends reality. And to create this feedback, the pursuit of graphics quality is inevitable, which is why video games have always emphasized the optimization of graphics quality, and players have been rushing to it. Moreover, in some games, the background is often ignored by developers, but these contents are easy to be noticed by game players, but it is difficult for developers to design carefully everywhere, so neural networks are needed to help developers complete this part.

Generative Adversarial Networks (GAN) are a neural network architecture for generative models [6]. Generative models are models that are used to generate new cases based on existing samples, for example, a new set of similar but slightly different photos based on an existing set of photos. Currently the main applications of GAN are in generating face photos, image restoration, text-to-image transformation, etc [7]. Today's existing GAN applications are all about work and life, and as society has gradually started to replace people's jobs with machines, people now have more time for themselves, which is why the gaming industry has become a hot commodity. And what makes games different from the real world is that the images in games are often fictional and do not exist in the real world. And because the architectural background of the scene in the game and the actual building are not matched, Cycle GAN is not very dependent on the matching dataset [8]. For these reasons this paper optimizes the processing of images in games by using Cycle-GAN. Cycle-GAN is very effective in transforming the style of unpaired data sets, and it is possible to optimise game images without the limitations of the data set. The aim of this paper is to apply GAN to video games to give players a more realistic game world and help people to enjoy the game better.

The first part of the paper describes the design and implementation of the network, including the pre-processing of the dataset, the construction of the generator and discriminator, and the training process. The second part describes the problems in the network and the ways to improve it. The final part of the paper presents the achievements of the network and future improvements and suggests possible applications in other areas.

2. Method

The reason for using Cycle-GAN for optimization is that in-game and real-world architectural images are not paired, and Cycle-GAN is good at doing style conversions on unpaired data-sets, and using Cycle-GAN allows the display style and game style to be converted to each other without the limitations of the data-set [9].

2.1. Datasets and preprocessing

The entire data-set is made up of two unpaired data-sets called facades from Kaggle[10]. Representative examples are shown in Figure 1. The X data-set consists of 400 images, which are real-world images, and the Y data-set also contains 400 images, which are images of game backgrounds.

As for the pre-processing of the dataset, *d*uring the construction of the project, the data-set was read and then pre-processed to make it more suitable for training the network [11]. The images were for three channels and I normalized each channel to a normal distribution with a mean of 0.5 and a variance of 0.5.

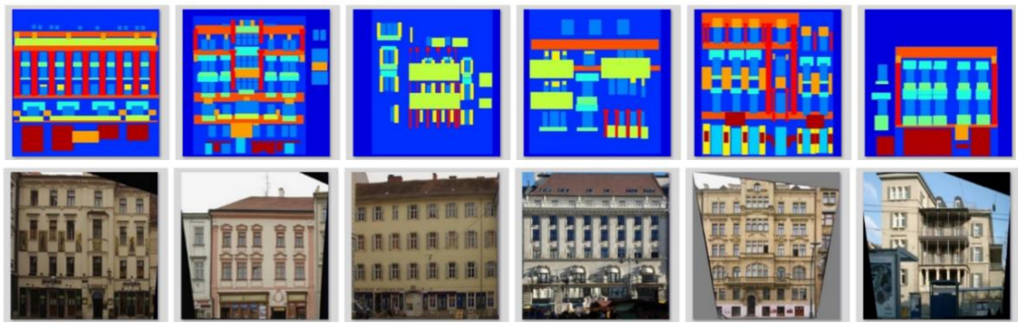

Figure 1. In this case simply justify the caption so that it is as the same width as the graphic.

2.2. Models

After constructing the data-set, two generators G, F are constructed (G generator converts images from X into the style of Y and F generator converts images in Y into the style in X), two discriminators M, N are constructed (M discriminator discriminates false images generated by X and N discriminator discriminates false images generated by Y) as shown in Figure 2. Adam is used as the optimizer because the network is large and the training time is long, in order to more easily find a more suitable set of hyper-parameters, so SGD was not applied as the optimizer. When building the generator and discriminator, mainly used convolutional blocks, which has the advantage of reducing the number of training parameters and preventing the network from becoming too large and causing slow training [12]. The structure of the generator is similar to that of the Encoder-Decoder, where the number of channels is first down-sampled to increase the number of channels and the height and width are reduced to get a "bottle", and then up-sampled to decrease the number of channels and increase the height and width to restore the size of the image to that of the generator, so that a new image can be obtained. The structure of the discriminator is simpler, just use the convolution block to reduce the height and width, and finally reduce the number of channels to 1. When predicting, each value of the height and width is averaged and used as the prediction value for the image.

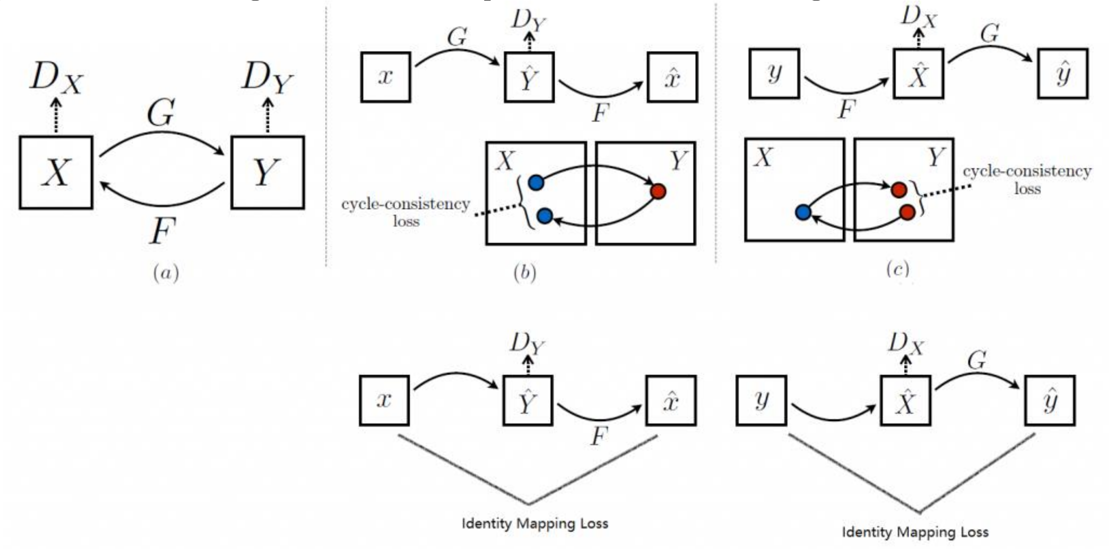

Figure 2. Structure of Cycle-GAN [8].

The core of the generator is computed using convolutional blocks with a logic similar to that of a self-encoder [13], using sub-samples to compress the image x in the data-set X from a large image with a few channels into a small image with many channels, and then using up-samples to expand the image to obtain the image G(x), using only convolutional blocks to increase the number of channels at the start of the computation, without changing the shape of the model.

The discriminator progressively increases the number of channels and decreases the height and width by convolution blocks, thus obtaining a multi-channel, small height, and width value. Finally, the probability of each image being judged as true is obtained by simply averaging the values.

2.3. Training

The training process is demonstrated in Figure 3. When training, firstly read the data X and Y in the data-set, and use generators G and F to generate fake images x' (fake images generated by x with the style of y), y' (fake images generated by y with the style of x), put the images into discriminator M and N to get the prediction results, and then put the fake images x ', y' into the generator F, G to get x" (with the fake picture x' as input, after one loop through the generator F, the resulting x' ', which is used to compare with the true picture x to calculate the cycle loss), y" (same as x"). With these structures in place, it is possible to calculate the generator loss, discriminatory loss, and cycle loss, weight each of these three and sum them to obtain the total loss, then back propagate and gradient descent can be used.

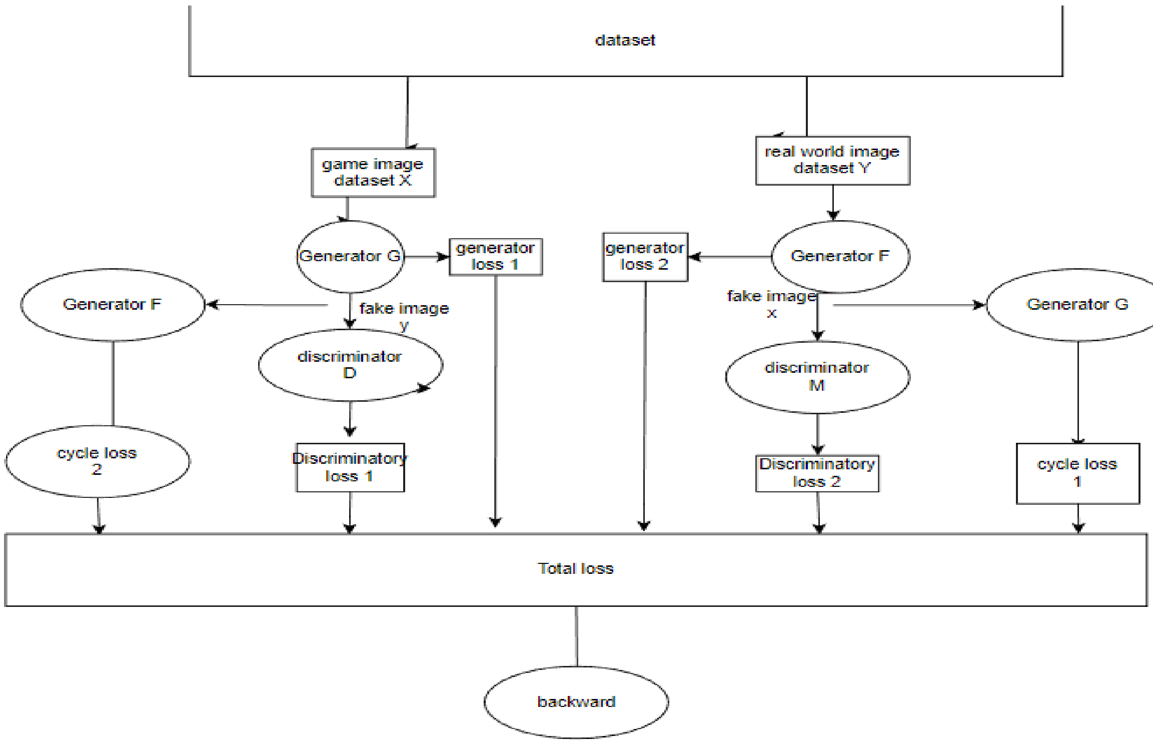

Figure3. The workflow of the network.

3. Result

3.1. Result of the original cycle-GAN

The loss values of various loss functions are demonstrated in Figure 4. The generator works well because the discriminator is basically around 0.4 for it, indicating that the discriminator is not too sure whether the fake images are fake or not.

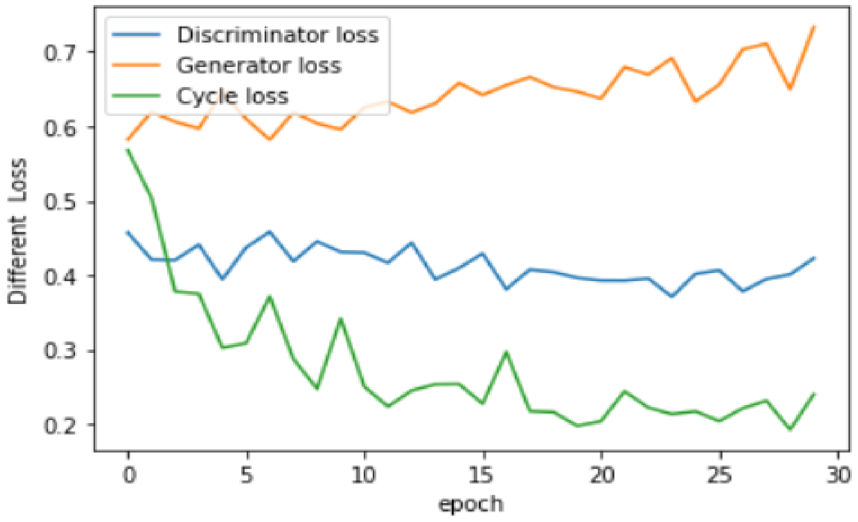

Figure 4. Loss values at various epochs.

And as can be seen from the Figure 4, the generator's loss doesn't go up much, it basically oscillates around 0.65, which is good because the generator and the discriminator are trained at the same time, and the discriminator is constantly optimizing, while the generator is also constantly optimizing, just because the discriminator is optimizing faster than the generator.

3.2. Result of improved model

When using this network, the network focuses on the entire image, migrating all the objects in the image in style, which leads to some weird behaviour, resulting in not only changing the skin of the horse to black and white stripes, but also the skin of the man riding the horse to a zebra [14]. And in this application, this problem also occurs. For example, treating people in the background of a game as part of a building, treating someone as a pillar and being converted into a realistic picture, resulting in weird results. Therefore, the network needs to be made aware of which parts are the focus and which parts are the noise and need its attention [15].

When doing natural language processing, a word in a sentence is often not independent and contextually relevant, but has a different relevance to different words in the context [16], so when dealing with the word, it is important to see it in context and also to pay more attention to words that are more relevant to it, which uses what is often called the self-attentive mechanism. Therefore, each pixel in the image is also not independent, but contextually relevant., but the relevance is also different [17]. In this application we expect the network to see the whole building globally, but then ignore some of the noise information on it. For example, keeping the network's attention focused on the building and not dissipate it on some of the game characters. The self-attention mechanism can therefore achieve this and thus improve the network's shortcomings.

Since the Cycle-Gan of baseline is flawed in that it is very easy to add something non-subject to the subject [18], in order to keep the network focused on the subject, and since the attention mechanism is often used in networks of the Encoder-Decoder system, which is exactly what the generator is, taking the attention mechanism, an algorithm that acts on the convolutional blocks inside the generator might be a good choice.

The self-attentive mechanism is used, and this block was added to use the attention mechanism in the convolutional network. Where gamma is 0, just the original image is output and gamma is the parameter that can be learned. The network structure is: input the image, get the result of query and key, let the two results be dotted, use SoftMax to get the attention [19], then calculate the value, use the attention and value to dot the product, add the result obtained and the original image, to get the attention-weighted image. This image is then used to perform a convolution operation to complete the use of the attention mechanism. By using the attention mechanism, the image has a more appropriate value, so that it can be trained better in the next network implementation.

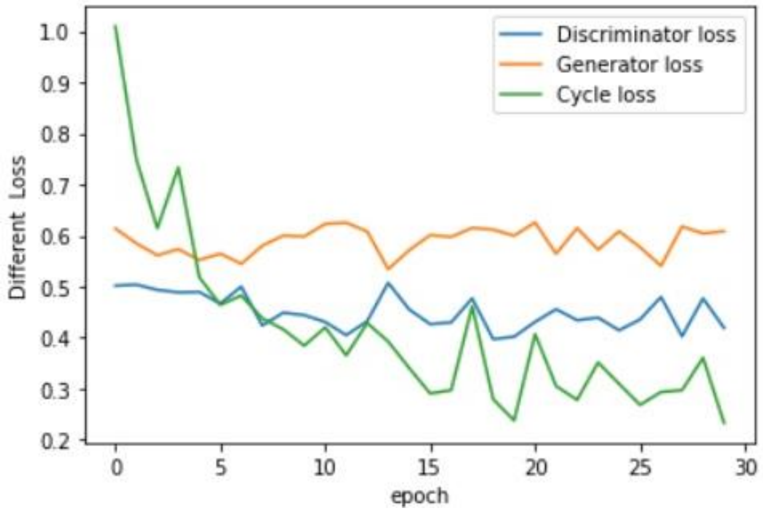

Figure 5. Network's loss graph after update.

From the results shown in Figure 5, the loss function of the generator has been fluctuating around 0.6, which is actually a good effect, because in the process of generator optimization, the discriminator has also been optimized, and during the training process, the result of the discriminator to determine whether the fake picture is a real picture has been around 0.45, indicating that it is difficult for the discriminator to judge the result generated by the generator, indicating that the attention mechanism can work on this part This indicates that the attention mechanism can work on this part, allowing the model to ignore some noise on the basis of insight into the global picture, thus generating a more realistic and realistic picture. The cyclic loss keeps decreasing, indicating that the generator generates images that are essentially structurally identical to the original image.

3.3. *Visualization result*

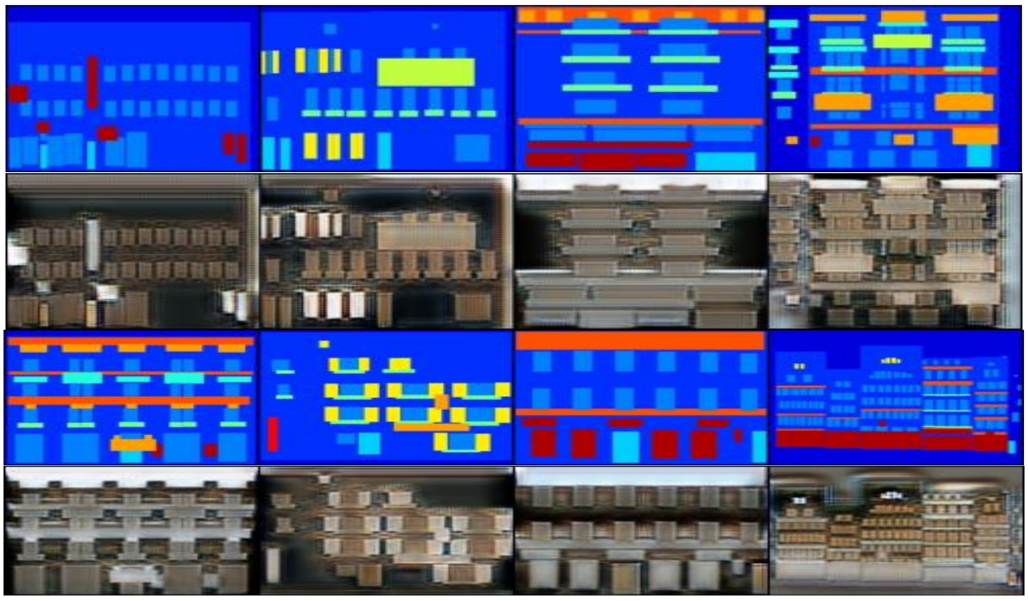

Figure 6. Structural images and generated realistic style images.

The results derived from the images in Figure 6 show that after Cycle-GAN training and generation, the original game images can be optimally processed to closely resemble real-world building images.

Cycle GAN did not change the overall architecture of the game's architectural background, but changed its style to become more realistic. So, it can bring comfortable experience for players.

4. Conclusion

Through the study, implemented an operation to optimize game images using Cycle-GAN. After the dataset is pre-processed, the data is trained by using generators and discriminators, and by adding a self-attentive mechanism to focus the network's attention on generating images of buildings. From the final generated image results, after Cycle-GAN training and generation, the original game images can be optimally processed to be close to the real-world building images. This research fills a gap in the field of GAN in gaming, allowing GAN to be applied not only to the real world but also to the virtual world in games. This research can help gamers to improve the quality of their games, enhance their immersion in their games and give them a better gaming experience.

But at the same time, the accuracy of the network needs to be further improved. By looking at the structure of Cycle-GAN, we can see that there are four networks to be optimized during training (two discriminators and two generators). If we want these networks to be more accurate, we would normally make them deeper and larger, but many networks in Cycle-GAN makes the training process very slow and causes the network to take a lot of time. The next step in optimizing Cycle-GAN is to speed up convergence, for example by using Batch normalization or other methods to speed up convergence, so that Cycle-GAN can be made more efficient.

On the other hand, image optimization can be applied not only to game images but also to other fields, for example, in medicine for X-ray images, Cycle-GAN can make the images clearer, which can help doctors to better analyse the X-ray images to have a more accurate judgment of the patient's condition. Cycle-GAN can also be used to optimize the clarity of images, as old photographs are one of the spiritual treasures of people, and their restoration is of great importance to people. Cycle GAN can also convert sketches into real pictures, so that users can get their expectations roughly in the shortest time. Cycle GAN is equivalent to visualizing the ideas in users' minds, making them easier to communicate.

References

[1] McAllister, G., & White, G. R. (2015). Video game development and user experience. In Game user experience evaluation, 11-35.
[2] González Sánchez, J. L., Padilla, Z. N., & Gutiérrez, F. L. (2009). From usability to playability: Introduction to player-centred video game development process. In International Conference on Human Centered Design, 65-74.
[3] Sweetser, P., & Wiles, J. (2002). Current AI in games: A review. Australian Journal of Intelligent Information Processing Systems, 8(1), 24-42.
[4] Zackariasson, P., Walfisz, M., & Wilson, T. L. (2006). Management of creativity in video game development: A case study. Services marketing quarterly, 27(4), 73-97.
[5] Dehghani, M., Montazeri, Z., Saremi, S., Dehghani, A., et al. (2020). HOGO: Hide objects game optimization. Int. J. Intell. Eng. Syst, 13(10), 216-225.
[6] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 1125-1134.
[7] Wang, X., & Gupta, A. (2016). Generative image modeling using style and structure adversarial networks. In European conference on computer vision, 318-335.
[8] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision, 2223-2232.
[9] Engin, D., Genç, A., & Kemal Ekenel, H. (2018). Cycle-dehaze: Enhanced cyclegan for single image dehazing. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 825-833.

[10] Suyash D. (2019). An unpaired-images dataset for training CycleGANs, URL: https://www.kaggle.com/suyashdamle/cyclegan
[11] Taleb, I., Dssouli, R., & Serhani, M. A. (2015). Big data pre-processing: A quality framework. In 2015 IEEE international congress on big data, 191-198.
[12] Roy, A. G., Navab, N., & Wachinger, C. (2018). Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. IEEE transactions on medical imaging, 38(2), 540-549.
[13] Feiniu, Y., Lin, Z., Jinting, S., Xue, X. I. A., & Gang, L. I. (2019). A review of the theory and application of self coding neural networks. Acta Sinica Sinica, 42(001), 203-230.
[14] Liu, L., Xi, Z., Ji, R., & Ma, W. (2019). Advanced deep learning techniques for image style transfer: a survey. Signal Processing: Image Communication, 78, 465-470.
[15] Cho, K., Courville, A., & Bengio, Y. (2015). Describing multimedia content using attention-based encoder-decoder networks. IEEE Transactions on Multimedia, 17(11), 1875-1886.
[16] Chowdhary, K. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649.
[17] Anokhin, I., Demochkin, K., Khakhulin, T., Sterkin, G., Lempitsky, V., & Korzhenkov, D. (2021). Image generators with conditionally-independent pixel synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 14278-14287.
[18] Chen, Y., Lai, Y. K., & Liu, Y. J. (2018). Cartoongan: Generative adversarial networks for photo cartoonization. In Proceedings of the IEEE conference on computer vision and pattern recognition, 9465-9474.
[19] Wang, F., Cheng, J., Liu, W., & Liu, H. (2018). Additive margin softmax for face verification. IEEE Signal Processing Letters, 25(7), 926-930.