CyberPunkGAN:
Generative Adversarial Networks - Turning City into Cyberpunk

Gary Li
Hong Kong University of Science and Technology
khlibg@ust.connect.hk

Abstract

In this paper, I proposed a solution to transforming street views and buildings photos into neon city style or so called CyberPunk style images. My solution is a learning-based method that many literatures have implemented it in different kind of style transfer, espically in cartoonization. By extracting high-level features from the orginal contents using pre-trained network VGG, the generated images are able to keep high-level of details from the input photos. Based on this, we also added the few losses like grayscale style loss, grayscale adversarial loss, and color reconstruction loss, so that the model not only able to keep most of the content features from input photos, but also able to transfer the color tone close to the style images. The CyberPunkGAN is trained with unpaired data which do not require any labelling on training data. The model takes content photos and style images as input, by adding new losses on the backbone Generative Adversarial Network (GAN). The result images show that comparing to modern text-to-image models e.g. *DALL-E and Stable Diffusion, our model still returns results with higher level of details as we use real photos as input. Therefore, the our result provides a much sharper edges of its objects, much clearer, and more realistic.*

1. Introduction

Cyberpunk, originally it was a subgenre of science fiction in 1960s to 1970s that refers to a dystopian futuristic setting. With more and more cyberpunk style creative works are produced in recent years, cyberpunk style contents go viral very quickly, espically after the released of the console game "CyberPunk 2077".

Since then, cyberpunk has incredible influence on different types of media such as gaming, anime, movie, graphic design, and also other creative contents. Many artists and creators are invloving in creating cyberpunk style images and videos, and people love it. Therefore, it has moved from a subculture to a mainstream style.

(a) Original Street View (b) Model Result

Figure 1. An example of CyberPunk stylization. (a) A photo of street view from Hong Kong. (b) Result image generated by the model that transforms the photo (a) into CyberPunk style.

To create a cyberpunk style image, those image creators usually draw it from stratch, or use other pictures as reference to re-create it, which invloves many manual works. For artists, creating high-quality cyberpunk style pictures requires careful consideration of the texture, color, brightness of neon light, and the light performance on edges of buildings. As Cyberpunk is getting popular, cyberpunk and neon lights filters are exploding in social media, yet the existing in social media are usually just using simple color filters to attract users by the gimmick.

My original purpose was turning Hong Kong street into CyberPunk style by using mobile phone app. But for the scope of this project, the result image generation is only limited to python script. Therefore, in this project, I will propose a solution to transforming real life pictures into cyberpunk style images using Generative Adversarial Network to perform style transform. In this way, the model can allows normal people who are not proficient in art to apply cyberpunk style on their own photos.

My method is using Generative Adversarial Networks (GANs) [7] by putting unpaired photos and cyberpunk style images into training, therefore no labelling is required. To improve the neon visual effects of the generated images, I

applied three efficient loss functions from the literature [1]. The three loss functions are as the followings: the grayscale style loss, the color reconstruction loss, and the grayscale adversarial loss, which the first two are related to generative network, and the third one is related to discriminative network. In generator of our GAN model, the grayscale style loss and the color reconstruction loss can turn the generated image towards neon light style and preserve the color of the photos. The grayscale adversarial loss in the discriminator can make the color of generated image more vivid. In the discriminator network of the model, I also used the edge-promoting adversarial loss [2] to preserve clear edges.

In order to keep the content of the original photos when generating new images, pre-trained model VGG19 [16] is applied as the perceptual network to obtain the ℓ_1 loss of the deep perceptual features of the generated style images and original photos. Before CyberPunkGAN starts training, we need to smooth the edges of content images and perform an initialization training only on the generator to make the training process much more easier. The initialization can reduce the total training time as the training will be more stable and faster to converge.

This paper follows in the order of section 2 with related literatures this paper is referencing, section 3 on how to perform the data collection, section 4 on how the model architecture is constructed and the details of loss functions, section 5 on the results of CyberPunkGAN comparing other state-of-the-art text-to-image models, and finally section 6 on short summary of this paper and some possible future works could be done.

2. Related Work

2.1. Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) [7], introduced in 2014, is making using of an generator to generate fake data, and feed the fake data into the discriminator that distinglishs the fake data from real data.

The popularity of GAN has been raising because of the ability to generate new data based on the training data. Since 2014, many researchers and engineers have been developing fantastic GAN models into real life applications such as photorealistic images creation [13], text-to-image [18], image-to-image translation [19], face aging [5], super-resolution images creation, etc. The newest text-to-image technology used by DALL · E 2's development team called VQGAN+CLIP [4], is also making use of GAN.

2.2. Image Style Transfer

Image Style Transfer [6] introduced a method to split out content representation and style representation of Convolutional Neural Network (CNN) [10]. It proposed that in different levels of CNN, the information contained are different. In general, the higher layers of the CNN network preserve high-level contents of an image, while the lower layers of network contains more detailed pixel information. By this method, we can extract the content representation from the real life photos, and extract style representation from the cyberpunk style photos, and let the model to learn these two sets of images.

2.3. CartoonGAN

This literature [2] introduced a solution to transfrom photo of real-world scenes into cartoon style images with high quality. At the moment this paper is released, it outperformed that state-of-the-art methods in generating cartoon style images based on style transfer of content images. The main ideas this paper are using semantic loss from VGG networks, adding edge-promoting adversarial loss for preserving edges, and on top of them proposed initialization phase to improve the convergence.

2.4. AnimeGAN

This literature [1] is using CartoonGAN [2] as backbone, further improve the model cartoonization by proposing novel lightweight generative adversarial network. This network proposed three losses, which are grayscale style loss, grayscale adversarial loss and color reconstruction loss. By reconstructing the generator and putting new parameters into loss function, AnimeGAN is able to outperform other models and train the model with lower memory capacity.

3. Dataset

For this model, we do not have a standard and available image dataset available online. By searching Google and some photography providers *e.g.* iStockphoto and Pinterest, there are lots of suitable images for usage. As there may have copy right issues on data sources, the websites mentioned above are not allowed to use or require high payments. With limited budget, I decided to retrieve the data from Pexels.com API and other free sources that allows usage of images for academic use. I collected the content image (street view) by using the search key 'Hong Kong Street', 'Hong Kong Buildings', 'Street view' and other related combinations of above. But the data are from free API and sources, causing the quality of images are unstable and high variations, for example some of them contains human faces or bodies, animals' pictures, and other unrelated pictures that are not easy to complete the cleaning with only human, face, or object detection models; therefore, data selection must be done manually.

And for the cyberpunk style images, searching the images by the search key of 'Cyberpunk', 'Cyberpunk City' are not giving valid results. For example, the results from

(a) Valid content image

(b) Invalid content image

Figure 2. Examples of valid and invalid content images that may gathered from the free sources.

(a) Valid style image

(b) Invalid style image

Figure 3. Examples of valid and invalid style images that may gathered from the free sources.

Pexels.com are giving out real street view instead of CyberPunk style images. Using 'Neon city' as search key will give out more related results.

The content data are selected if the photo contains clear street views or building views. The style images are selected if the images are in cyberpunk city or neon city style; The selection process is subjective as we do not have structured selection methods to filter the correct style images for training. In total, I have collected 7915 training data for content images with 6000 of them as training data, also 752 cyberpunk style images from the sources. 752 of CyberPunk style images are not enough for training, yet, due to very limited style images available, we will use 752 style images for training.

4. Model

The GAN framework has two CNNs, with one as the Generator G and another one as the Discriminator D. By feeding the real life image into the Generator G, it will produce a fake cyberpunk image, followed up by the Discriminator D, which will judge whether the image generated is a real cyberpunk image. The objective of the model is to find the weights (G^*, D^*) of G and D that minimize the minmax loss function [1, 2, 7]:

$$(G^*, D^*) = \arg\min_G \max_D L(G, D) \tag{1}$$

4.1. CyberPunkGAN Architecture

The generator of CyberPunkGAN is a symmetrical encoder-decoder network, composed with some convolutions and different sub-networks. First, let's introduce the sub-network existing in the Generator.

In Fig.4, Conv-Block is composed of a standard convolution with kernal size of 3×3, followed by instance normalization layer [17], and the Leaky-ReLU activation function [11]. The negative slope of Leaky-ReLU is setting 0.2.

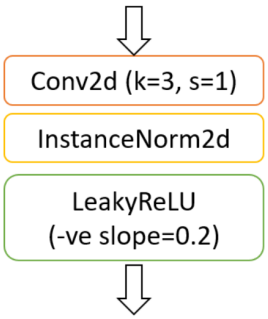

Figure 4. Conv-block

In Fig.5, DSConv [3] is similar with Conv-Block, and is composed of depthwise separable convolution with kernal size of 3×3, followed by instance normalization layer, and the Leaky-ReLU activation function. And then, a Conv-Block is added afterwards with both kernal size and stride are 1.

In Fig.6, the inverted residual block (IRB) [15] contains Conv-Block with 512 feature maps, followed by depthwise convolution with kernal size of 3×3, instance normalization layer, and the Leaky-ReLU activation function. Then, it decreases the feature maps to 256 in a standard convolution with both kernal size and stride are 1. Finally, $\bigoplus$ represents element-wise addition. Compared to the normal residual block [8], IRB can reduce the parameters and computation time of the model.

In Fig.7 and Fig.8, these two are Down-Conv and Up-Conv. H and W represents height and width respectively.

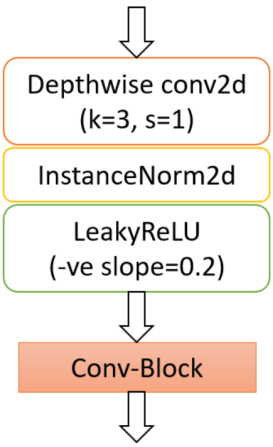

Figure 5. DSConv

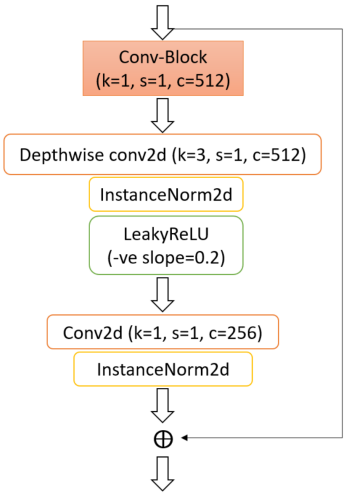

Figure 6. Inverted Residual Block (IRB)

They both contains DSConv, and Resize layer that reduces or increases double the size. Down-Conv is proposed to replace max-pooling, so that it can reduce the resolution of feature maps without lossing the feature information. It adds up the result from DSConv with kernal size of 3×3 and stride 2, and the result from DSConv with kernal size of 3×3 and stride 1 after halfing its feature maps size in Resize layer.

For Up-Conv in Fig.8, it is proposed to replace fractionally strided convolutions with stride $\frac{1}{2}$, so that it can increases the resolution of feature maps without causing checkerboard artifacts [14] which affects the quality of images. It is just simply doubling the feature maps and feed into DSConv with kernal size of 3×3 and stride 1.

As shown in Fig.9, the whole generator is composed by the sub-networks described above. It can be devided into 3

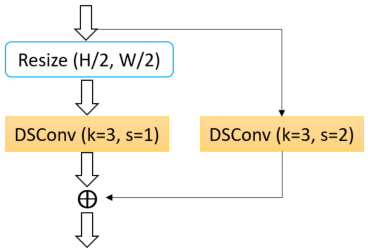

Figure 7. Down-Conv

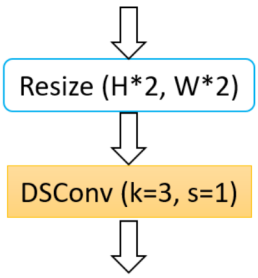

Figure 8. Up-Conv

parts: Encoder, Residual Blocks, and Decoder.

1. Encoder
   When we feed the input into the model, it first encounters the encoder. The encoder is composed by 2 Conv-Block with 64 output channels, followed by 1 Down-Conv, 1 Conv-Block, 1 DSConv with 128 output channels, and finally 1 Down-Conv, 1 Conv-Block with 256 output channels.

2. Residual Blocks
   Instead of using standard residual blocks, we are using 8 consecutive IRBs to reduce the number of parameters so that the training time is also reduced.

3. Decoder
   What the decoder is doing, is similar to reverting back the steps did by the encoder. It is composed by 1 Conv-Block with 128 output channels, followed by 1 Up-Conv, 1 DSConv, 1 Conv-Block with 128 output channels, and finally 1 Up-Conv with 128 output channels and 2 Conv-Block with 64 output channels. After that, $Tanh$ function is applied after going through a standard convolution with 3 output channels and the data size is same as the iuput layer.

For the Discriminator in Fig.10, it is relatively simple as it did not contain any special blocks or sub-networks. First, it changes the input by using standard convolution layer with kernal size 3×3, stride 1, and 32 channels, followed by a Leaky-ReLU activation function. Then, a format of

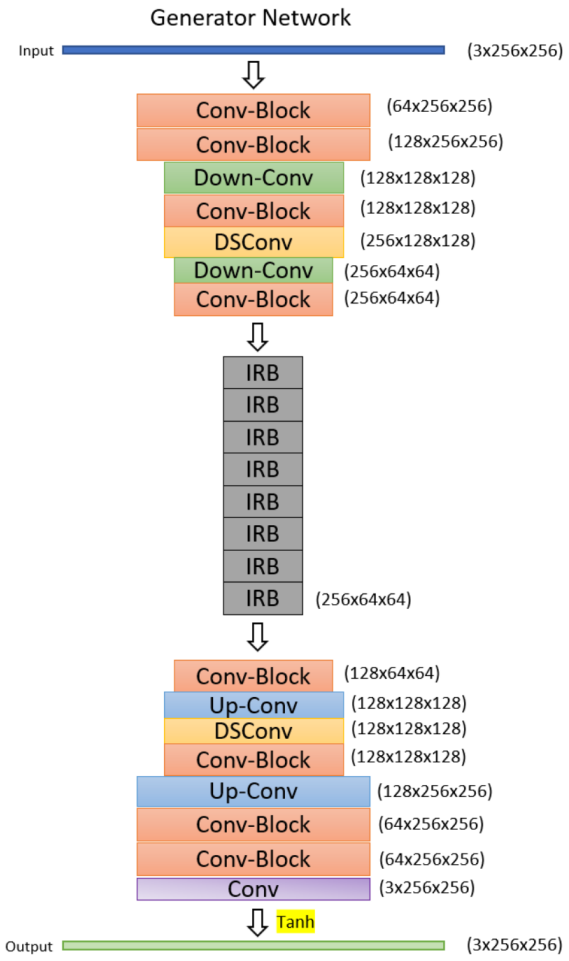

Figure 9. Generator Network

the order: 1 standard convolution with kernal size 3×3 and stride 2, Leaky-ReLU activation, another standard convolution with double the number of channels with kernal size 3×3 and stride 1, instance normalization layer, and another Leaky-ReLU activation, is repeated 3 times and each time the channels number is increased by 4 times. After that, 1 standard convolution with kernal size 3×3 and stride 1, instance normalization, Leaky-ReLU activation are applied. Finally, one single convolution with kernal size 3×3 and stride 1 is applied and the number of channels is decreased from 2048 to 1.

4.2. Loss Function

Let the real photo domain be P, and the style image domain be A. So that $S_{data}(p) = \{p_i | i = 1, \ldots, N\} \subset P$, $S_{data}(a) = \{a_i | i = 1, \ldots, M\} \subset A$, where N is the numbers of the real photo in the training set and M is the style images in the training set. Based on these, two new domains are generated.

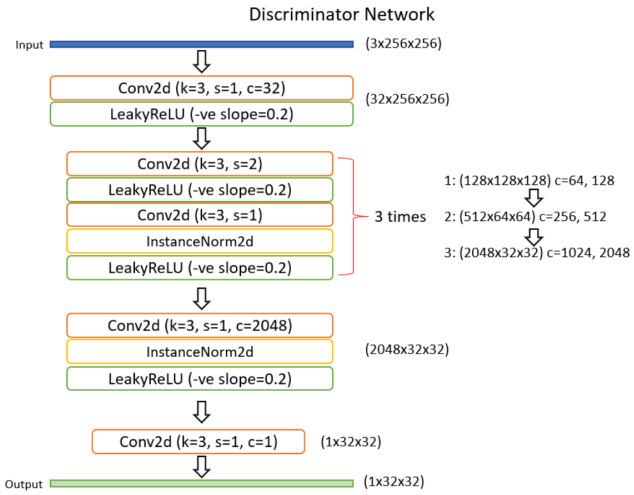

Figure 10. Discriminator Network

The first one is $S_{data}(x) = \{x_i | i = 1, \ldots, M\} \subset X$, where X is the transformation result of turning the style images $S_{data}(a)$ into grayscale to elminate color interference. The second one is $S_{data}(y) = \{y_i | i = 1, \ldots, M\} \subset Y$, where Y is the transformation result of turning the style images $S_{data}(a)$ into grayscale after removing edges.

Let G be the generator and D be the discriminator, the loss function $L(G, D)$ employed by least squares loss function LSGAN [12] can be expressed as follows:

$$\begin{aligned} L(G, D) = w_{adv} L_{adv}(G, D) + w_{con} L_{con}(G, D) \\ + w_{gra} L_{gra}(G, D) + w_{col} L_{col}(G, D) \end{aligned} \tag{2}$$

where $L_{adv}(G, D)$ is the adversarial loss that affects style transformation process in G, $L_{con}(G, D)$ is the content loss which helps to retain the content of inputs photo, $L_{gra}(G, D)$ is the grayscale style loss which makes clear cyberpunk style on textures and lines, $L_{col}(G, D)$ is the color reconstruction loss which helps to color.

For $L_{con}(G, D)$ and $L_{gra}(G, D)$, VGG-19 [16] is used as perceptual network to extract high-level semantic features of the images. To ensure the generated images retain semantic content from the input content photos, the 4^{th} layer of block 4 in VGG-19 is applied into loss function. These two terms can be expressed as:

$$\begin{aligned} L_{con}(G, D) = E_{p_i \sim S_{data}(p)} [\| VGG_l(p_i) \\ - VGG_l(G(p_i)) \|_1] \end{aligned} \tag{3}$$

$$\begin{aligned} L_{gra}(G, D) = E_{p_i \sim S_{data}(p)}, E_{x_i \sim S_{data}(x)} \\ [\| Gram(VGG_l(G(p_i))) \\ - Gram(VGG_l(x_i)) \|_1] \end{aligned} \tag{4}$$

, where l refers to the feature maps of l th layer in VGG-19 (We are using conv4-4 of VGG-19 network)

For $L_{col}(G, D)$, in order to make the image color reconstruction better, we convert the image color in RGB format to YUV format. ℓ_1 loss is applied to Y channel, Huber Loss is applied to U and V channels. The loss can be defined as:

$$\begin{aligned} L_{col}(G, D) = E_{p_i \sim S_{data}(p)}[&\|Y(G(p_i)) - Y(p_i)\|_1 \\ &+ \|U(G(p_i)) - U(p_i)\|_H \\ &+ \|V(G(p_i)) - V(p_i)\|_H] \end{aligned} \tag{5}$$

, where $Y(p_i), U(p_i), V(p_i)$ represent the 3 channels of YUV in image p_i, and H represents Huber loss.

Finally, the loss function of generator can be expressed as follows:

$$\begin{aligned} L(G) = &w_{adv} E_{p_i \sim S_{data}(p)}[(G(p_i) - 1)^2] \\ &+ w_{con} L_{con}(G, D) \\ &+ w_{gra} L_{gra}(G, D) \\ &+ w_{col} L_{col}(G, D) \end{aligned} \tag{6}$$

For the loss function of discriminator, both edge-promoting adversarial loss [2] and novel grayscale adversarial loss [1] are applied to help the model generating color images instead of grayscale image.

$$\begin{aligned} L(D) = &w_{adv} E_{a_i \sim S_{data}(a)}[(D(a_i) - 1)^2] \\ &+ E_{p_i \sim S_{data}(p)}[(D(G(p_i)))^2] \\ &+ E_{x_i \sim S_{data}(x)}[(D(x_i))^2] \\ &+ 0.2 E_{y_i \sim S_{data}(y)}[(D(y_i))^2] \end{aligned} \tag{7}$$

,where $E_{y_i \sim S_{data}(y)}[(D(y_i))^2]$ is the edge-promoting adversarial loss with scaling of 0.2 is applied to prevent sharp edges, $E_{x_i \sim S_{data}(x)}[(D(x_i))^2]$ is the grayscale adversarial loss.

4.3. Training

As mentioned above, I have collected 7915 training data for content images with 6000 of them as training data and 1915 of them as testing data, and also 752 cyberpunk style images all for training. No testing data on style images as we only require the content images and the generator to perform model testing. All the data are resized into 256×256, and the style images are smoothed by removing edges from images afterwards. Before the real training, the generator is pre-trained for 5 epochs with learning rate of 1e-3 and betas=(0.5, 0.999) using Adam optimizer [9] to have a better initialization, so that converges faster and more stable. After that, both generator and discriminator are trained together for 95 epochs. They both using Adam as optimizer with betas=(0.5, 0.999). But the learning rates are different that generator and discriminator are using 2e-4 and 4e-4 as learning rate respectively. The weights on different losses are initialized as $w_{con} = 1.5$, $w_{gra} = 3$, $w_{col} = 30$, $w_{adv} = 10$. As the training is time comsuming and the training resources are not enough for me to try other parameters, these are the final weights.

5. Results

As there is no existing model of using similar method to perform style transfer specificly in CyberPunk or neon city style, I will compare the result by using state-of-art image generation models *e.g.* DALL-E (Fig.11), Stable Diffusion (Fig.12). Therefore, there are no original photos to compare using those text-to-image models. But for our CyberPunkGAN model, we can compare the result with the inputs (Fig.13) and generated outputs (Fig.14). We can see, from the results of text-to-image models, their generated images are more cartoon, and the colors are unreal. The composition of the images are pretty good, we can even reference some of the them to the excat location in Hong Kong, yet the content information are distorted. In contrast, as we applied VGG-19 to obtain perceptual features from content photos, the generated results only cause only very little distortion. However, the style transfer still have some issue on the color reconstruction. As we can see from second image of Fig.14, the color of sky is unlike the other images that it is not perfectly constructed. This happen maybe due to not enough style images data for training.

6. Conclusions

In this paper, I implemented CyberPunkGAN to transfer the real streets and buildings view into CyberPunk style or neon city style. The researchs on Generative Adversarial Networks (GANs) are popular before and the number of related research is decreasing recent year. As we can see many new models and technique are created on top of GAN nowadays *e.g.* VQGAN+CLIP [4]. Although the min max loss property of GAN makes the model training unstable, there are lots of methods to stabilize the error or even fasten the training process. For example, this paper is taking CartoonGAN [2] and AnimeGAN [1] as references, as these two literatures provide very mature solutions on transforming real-life pictures or videos into cartoon style. However, the losses I take reference may not be perfect for this model as the base idea of Cartoon style transfer and CyberPunk style transfer are different. Cartoon images may need to emphasize on the edges in cartoon images, but CyberPunk images need to focus more on the color reconstruction instead as the neon light are not easy to deal with.

With limited resources, replicating the mature literatures and apply it into implementation is best thing I can do for the model so far. Therefore, if there is any extra resources, the future works would be the followings: First, collect more CyberPunk style data with at least 1200 images, 1500 to 2000 would be better. Second, develop a schema to fil-

Figure 11. Examples results generated by DALL-E

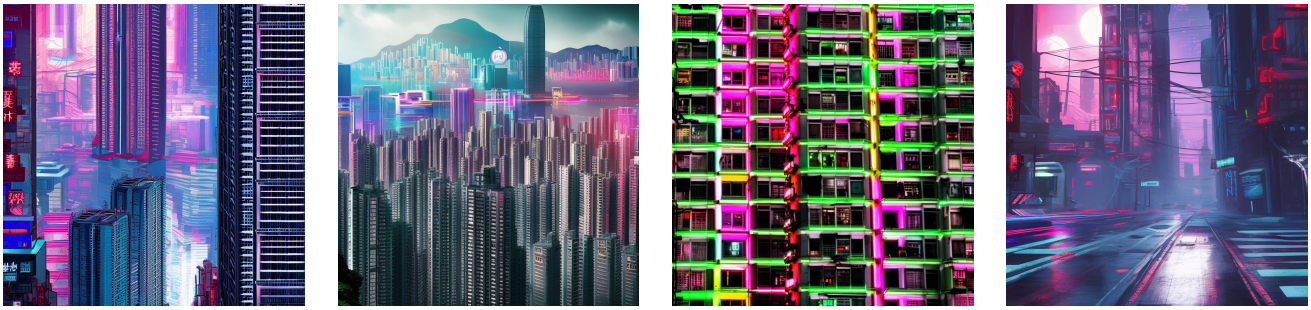

Figure 12. Examples results generated by Stable Diffusion

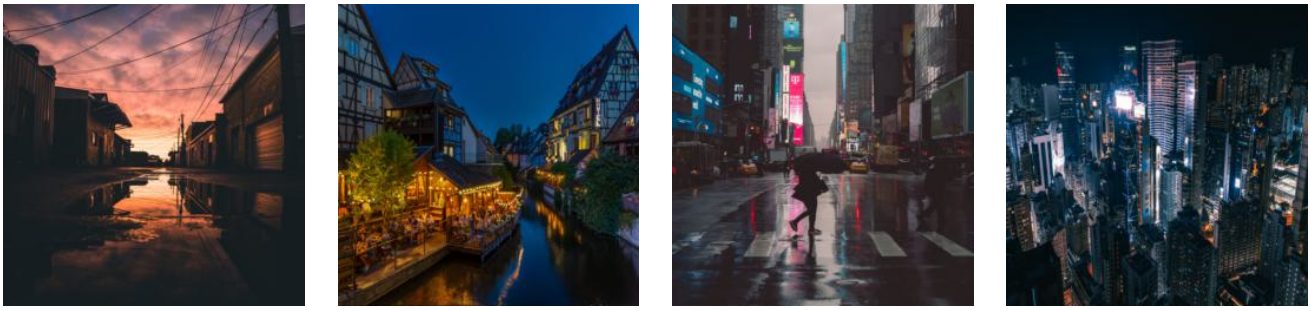

Figure 13. Original Inputs

Figure 14. CyberPunkGAN Model Outputs

ter the content data as judgement from only single person (me) is subjective. Third, create new losses that captures the neon lights better, perhaps using RGB instead of YUV is better. Fourth, reduce the size of the parameters if possible. Even though the number of parameters are already reduced by using IRBs, it taken 2 days to train the model with VM that contains 6 CPU cores, 112GB RAM, and a Tesla V100 GPU. Finally, save the last few models instead of single final model because the training of GAN is unstable, leads to dissimilar performance among models in different epochs even though it is just 1 or 2 epochs afterwards. If it is possible, a user interface is required no matter in jupyter(python)

scripy, webpage, or mobile app. In that way, size of the model is another concern if we need to apply the CyberPunkGAN on mobile phone.

References

[1] Jie Chen, Gang Liu, and Xin Chen. Animegan: A novel lightweight gan for photo animation. In Kangshun Li, Wei Li, Hui Wang, and Yong Liu, editors, *Artificial Intelligence Algorithms and Applications*, pages 242–256, Singapore, 2020. Springer Singapore. 2, 3, 6

[2] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9465–9474, 2018. 2, 3, 6

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016. 3

[4] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. Vqgan-clip: Open domain image generation and editing with natural language guidance, 2022. 2, 6

[5] Julien Despois, Fré déric Flament, and Matthieu Perrot. AgingMapGAN (AMGAN): High-resolution controllable face aging with spatially-aware conditional GANs. In *Computer Vision – ECCV 2020 Workshops*, pages 613–628. Springer International Publishing, 2020. 2

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 2

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1, 2, 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 6

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[11] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013. 3

[12] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, 2017. 5

[13] Nuriel Shalom Mor. Generating photo-realistic images from lidar point clouds with generative adversarial networks, 2021. 2

[14] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 4

[15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. 3

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 2, 5

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016. 3

[18] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks, 2016. 2

[19] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020. 2