Progressive GANs and Pix2Pix translation

Runchen Hu (rh2619) Shen Wang(sw3838) Lifan Wang(lw2435)

1. Introduction

In relation with image synthesis, it is pretty common to generate realistic, high-resolution and abundant types of images with some generic transformation network. A good generic network could capsulate different types of image transformation within only one trained neural network. However, there might be several problems. For example, the training process is not stable, the generator and the discriminator sometimes compete with each other unhealthily and unbalancedly. What's worse, the trade-off between resolution and variety of the generated images would cause our network structure becoming extremely complicated which will also take a long time to train. In order to obtain a better understanding of GAN, we as a group mainly focused on problems like how to modify or streamline the network structure and tried to get better results based our own knowledge. In this project, we developed unconditional GAN (Progressive Growing GAN) to generate realistic and high-resolution image (1024*1024). We also implemented conditional GAN (cGAN) to help retrieve the original ground truth image from a image segmentation map.

In the PG-GAN part, we reproduced [1] results on CelebA dataset, and then applied this training mechanism into CIFAR-10, analyzing whether the training process remains stable in different genres of samples. The results will be shown in the following part of this report. In the cGAN part, we started from the Pix2Pix framework. We first reproduced the Pix2Pix experiment by ourselves in several different datasets. Then we tried to optimize the depth and kernel size of the network structure by redesigning the U-Net and the PatchGAN for the discriminator. Finally we redefined the UNet structure by adding residual blocks in the generator, and hierarchically arranging them in reasonable order.

Evaluation metric is another important part of our paper, we applied the Multi-scale structure similarity into PGGANs, which is more convincing in capturing local and global features of high-resolution images than others in this case. We casted a manual evaluation experiment by sending out a questionnaire to get more comprehensive analysis. We also discussed and concluded the necessity to use some subjective measurements for the success rate of image-to-image translation.

2. Previous work/references

One traditional method in image synthesis is through variational auto-encoder (VAE), using L1 loss function, but people have witnessed some blurring in details of synthesized images comparing with real images. The other method, which is quite popular these years, is Generative Adversarial Nets (GANs), and it is behaving prominently in many applications, including generating realistic images and image-to-image translation. Thus, more and more researchers are trying to apply this new type of unsupervised learning method. Though some progress has been made, there are still some problems remaining.

2.1. Generative adversarial networks

In order to achieve high quality and fast training rate, GANs nowadays tend to implement progressive growing training mechanism instead of fixed layers. However sometimes it still would get stuck in unhealthy competitions, such as 'variation problem', when GANs capture only features from a subset of variation from the training data. Recent researches show orthogonalizing the feature vectors in a mini-batch, adding repelling regularizer, adding multiple generators and discriminators could to some extent resolve variation problem.

Meanwhile, GANs are prone to the escalation of signal magnitudes and covariate shift problem. In these cases, researchers usually use equalized learning rate and feature normalization to overcome it. As for evaluation, there are inception score, multi-scale structural similarity, birthday paradox, etc. The most commonly used one is MS-SSIM, but some researches argue that they are prone to small texture loss, and its performance is bad in training set similarity measurement.

2.2. Generate realistic images

Currently, the most prominent approaches to generate realistic images are autoregressive models [2], variational auto-encoders (VAE) [3], and generative adversarial nets [4]. VAEs are easy to train but it tends to produce blurry results due to restrictions in the model. Although recent works have improved its performance a lot, it is still not good enough. GANs produce sharp images, albeit only in fairly low resolutions and with somewhat limited variation, and the training continues to be unstable despite recent progress [5,6,7]. The idea of growing GANs progressively is related to the work of Wang [8], who use multiple discriminators that operate on different spatial resolutions. That work in turn is motivated by Durugkar [9] who use one generator and multiple discriminators concurrently, and Ghosh [10] who do the opposite with multiple generators and one discriminator. Hierarchical GANs [11,12,13] define a generator and discriminator for each level of an image pyramid.

2.3. Image-to-Image Translation

The goal of image-to-image translation is to translate an input image from one domain to another domain given input-output image pairs as training data, which is usually achieved by conditional GANs model. Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y, and y could be any kind of auxiliary information, such as class labels or data from other modalities.

In cGANs, compared with L1 loss, which people usually apply to measuring the success rate of translation though causing blurry effects, the adversarial loss performs better, which could be learned automatically by the network and recover some details from the ground truth images [14]. Even though adversarial loss is a better loss function, the output images sometimes are still in low resolution and still missing some details. That is why our new cGAN network used a hierarchical UNet for generator to synthesize the high-level details and low-level details separately. We also introduced a PatchGAN for discriminator to determine the local similarity between a synthesized image and a real one. In the following section, we would illustrate the power of that network by using the dataset of cityscapes, and by results of retrieving the cityscapes from a processed segmentation image.

3. Methods

GANs simultaneously train two models: a generative model G, which captures the data distribution, and a discriminative model D, which estimates the probability that a sample came from the training data rather than G [15]. The training procedure for G is to maximize the probability of D making a mistake. G is a generative network, we give a random noise z as input to generate an output image G(z). D is a discriminative network to judge if an image is real or fake. The input for D is x, which represent an image. The output is D(x) that means the probability of real image. If it's 1, it means it's a 100% real image, if it's 0, it means it's a 0% real image.

3.1. Progressive GANs

Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here N ×N refers to convolutional layers operating on N × N spatial resolution. This allows stable synthesis in high resolution training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

3.1.1. Objective

The formulation of GANs is the famous V(G, D) formula.

$$\min_G \max_D V(D,\ G) = \mathbb{E}_{x\sim Pdata}\left[logD(x)\right] + \mathbb{E}_{x\sim P(z)}[\log\left(1 - D(G(z))\right)]$$

The main idea of this loss function is to maximize the value of D(G(z)) (the probability of discriminator considers the image as true), and to minimize the value of D(G(z)), that is, to maximize the value of (1 - D(G(z))).

3.1.2. Network architectures

Generator:

G	Params	OutputShape	WeightShape
Latents_in		(?,512)	
Labels_in		(?,0)	
Lod		()	
4x4/PixelNorm		(?,512)	
4x4/Dense	4194816	(?,512,4,4)	(512,8192)
4x4/Conv	2359808	(?,512,4,4)	(3,3,512,512)
ToRGB_lod5	1539	(?,3,4,4)	(1,1,512,3)
8x8/Conv0_up	2359808	(?,512,8,8)	(3,3,512,512)
8x8/Conv1	2359808	(?,512,8,8)	(3,3,512,512)
ToRGB_lod4	1539	(?,3,8,8)	(1,1,512,3)
Upscale2D		(?,3,8,8)	
Grow_lod4		(?,3,8,8)	
16x16/Conv0_up	2359808	(?,512,16,16)	(3,3,512,512)
16x16/Conv1	2359808	(?,512,16,16)	(3,3,512,512)
ToRGB_lod3	1539	(?,3,16,16)	(1,1,512,3)
Upscale2D_1		(?,3,16,16)	
Grow_lod3		(?,3,16,16)	
32x32/Conv0_up	2359808	(?,512,32,32)	(3,3,512,512)
32x32/Conv1	2359808	(?,512,32,32)	(3,3,512,512)
ToRGB_lod2	1539	(?,3,32,32)	(1,1,512,3)
Upscale2D_2		(?,3,32,32)	
Grow_lod2		(?,3,32,32)	
64x64/Conv0_up	1179904	(?,256,256,64)	(3,3,256,512)
64x64/Conv1	590080	(?,256,256,64)	(3,3,256,256)
ToRGB_lod1	771	(?,3,64,64)	(1,1,256,3)
Upscale2D_3		(?,3,64,64)	
Grow_lod1		(?,3,64,64)	
128x128/Conv0_up	295040	(?,128,128,128)	(3,3,128,256)
128x128/Conv1	147584	(?,128,128,128)	(3,3,128,128)
ToRGB_lod0	387	(?,3,128,128)	(1,1,128,3)
Upscale2D_4		(?,3,128,128)	
Grow_lod0		(?,3,128,128)	
Images_out		(?,3,128,128)	
Total	22933394		

Discriminator:

D	Params	OutputShape	WeightShape
Images_in		(?,3,128,128)	
Lod		()	
FromRGB_lod0	512	(?,128,128,128)	(1,1,3,128)
128x128/Conv0	147584	(?,128,128,128)	(3,3,128,128)
128x128/Conv1_down	295168	(?,256,64,64)	(3,3,128,256)
Downscale2D		(?,3,64,64)	
FromRGB_lod1	1024	(?,256,64,64)	(1,1,3,256)
Grow_lod0		(?,256,64,64)	
64x64/Conv0	590080	(?,256,64,64)	(3,3,256,256)
64x64/Conv1_down	1180160	(?,512,32,32)	(3,3,256,512)
Downscale2D_1		(?,3,32,32)	
FromRGB_lod2	2048	(?,512,32,32)	(1,1,3,512)
Grow_lod1		(?,512,32,32)	
32x32/Conv0	2359808	(?,512,32,32)	(3,3,512,512)
32x32/Conv1_down	2359808	(?,512,16,16)	(3,3,512,512)
Downscale2D_2		(?,3,16,16)	
FromRGB_lod3	2048	(?,512,16,16)	(1,1,3,512)
Grow_lod2		(?,512,16,16)	
16x16/Conv0	2359808	(?,512,16,16)	(3,3,512,512)
16x16/Conv1_down	2359808	(?,512,8,8)	(3,3,512,512)
Downscale2D_3		(?,3,8,8)	
FromRGB_lod4	2048	(?,512,8,8)	(1,1,3,512)
Grow_lod3		(?,512,8,8)	
8x8/Conv0	2359808	(?,512,8,8)	(3,3,512,512)
8x8/Conv1_down	2359808	(?,512,4,4)	(3,3,512,512)
Downscale2D_4		(?,3,4,4)	
FromRGB_lod5	2048	(?,512,4,4)	(1,1,3,512)
Grow_lod4		(?,512,4,4)	
4x4/MinibatchStddev		(?,1,4,4)	
4x4/Conv	2364416	(?,512,4,4)	(3,3,513,512)
4x4/Dense0	4194816	(?,512)	(8192,512)
4x4/Dense1	513	(?,1)	(512,1)
Scores_out		(?,1)	
Labels_out		(?,0)	
Total	22941313		

Table 1: Generator and discriminator that we use with CELEBA-HQ to generate 1024×1024 images.

Table 1 shows network architectures of the full-resolution generator and discriminator that we use with the CELEBA-HQ dataset. Both networks consist mainly of replicated 3-layer blocks that we introduce one by one during the course of the training.

3.1.3. Training details

We train the networks using Adam with $\alpha = 0.001$, $\beta 1 = 0$, $\beta 2 = 0.99$, and $\epsilon = 10{-}8$. We do not use any learning rate decay or rampdown, but for visualizing generator output at any given point during the training, we use an exponential running average for the weights of the generator with decay 0.999. We use a mini-batch size 16 for resolutions $4^2 \rightarrow 128^2$ and then gradually decrease the size according to $512^2 \rightarrow 14$, $512^2 \rightarrow 6$, and $1024^2 \rightarrow 3$ to avoid exceeding the available memory budget.

3.1.4. Technique depth and innovation:

We increased the variation using mini-batch standard deviation. When doubling the resolution of the generator G and discriminator D, we let new layers fade in smoothly. Let's take the transition from 16 ×16 images to 32 ×32 images for example. During the transition we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. The toRGB represents a layer that projects feature vectors to RGB colors and fromRGB does the reverse, and both of them use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions. We also use equalized learning rate while training.

We deviate from the current trend of careful weight initialization, and instead use a trivial N(0, 1) initialization and then explicitly scale the weights at runtime. The benefit of doing this dynamically instead of during initialization is somewhat subtle, and relates to the scale-invariance in commonly used adaptive stochastic gradient descent methods such as RMSProp and Adam. Furthermore, we use pixel-wise feature vector normalization in generator to disallow the scenario where the magnitudes in the generator and discriminator spiral out of control as a result of competition, we normalize the feature vector in each pixel to unit length in the generator after each convolutional layer [1].

3.2. Pix2Pix translation

Pix2Pix uses a conditional generative adversarial network (cGAN) to learn a mapping from an input image to an output image, conditional GANs learn a mapping from observed image x and random noise vector z, to y, G: {x, z} → y. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes" [14].

3.2.1. Objective:

The objective of a conditional GAN can be expressed as

$$\mathcal{L}_{cGAN}(\mathrm{G}, \mathrm{D}) = \mathbb{E}_{x,y}[\log \mathrm{D}(\mathrm{x}, \mathrm{y})] + \mathbb{E}_{\mathrm{x},\mathrm{z}}[1 - \log \mathrm{D}(\mathrm{x}, \mathrm{G}(\mathrm{x}, \mathrm{z}))]$$

D(G(x, z)) means the probability that the image generated by G is real, G wants to make this probability as big as possible. That is to say G wants 1- D(G(x, z)) as small as possible, so we need to minimize the value of $\mathcal{L}$cGAN. With the growing ability of D, D(G(x, y)) should be as big as possible, so we need to maximize the value of $\mathcal{L}$cGAN.

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance [12]. The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense. We also explore this option, using L1 distance rather than L2 as L1 encourages less blurring. Our final objective is

$$G^* = \arg \min_G \max_D \mathcal{L}_{\mathrm{cGAN}}(\mathrm{G}, \mathrm{D}) + \lambda \mathcal{L}_{L1}(G)$$

As a matter of fact, the network can still learn a mapping from x to y without z, but the result could be deterministic outputs, and therefore fail to match any distribution other than a delta function. Instead, for our final models, we provide noise only in the form of dropout, applied on several layers of our generator at both training and test time. Despite the dropout noise, we observe only minor stochasticity in the output of our nets. Designing conditional GANs that produce highly stochastic output, and thereby capture the full entropy of the conditional distributions they model, is an important question left open by the present work [14].

3.2.2. Network architectures

We adapt our generator and discriminator architectures from those in [16]. Both generator and discriminator adopt similar architectures like Convolution-BatchNorm-ReLu, however we modify the network, let's discuss the details and improvement below.

3.2.3. Generator architectures(U-net)

Our purpose for the generator of image-to-image translation problem is to map the input to an output image,

the input and output differ in surface appearance, but both are renderings of the same underlying structure. The generator takes some input and tries to reduce it with a series of encoders (convolution + activation function) into a much smaller representation. The idea is that by compressing it this way we hopefully have a higher level representation of the data after the final encode layer.

Many previous solutions use the encoder-decoder network to solve the problem. In this network, Input passes through convolutional neural network as encoder that progressively downsample, until a bottleneck layer. Such a network requires that all information pass through all layers including the bottleneck. This method is a great deal of low-level information shared between input and output. To give generator a means to circumvent the bottleneck for information like this, we can add skip connections, that is "U-Net". The skip connections give the network the option of bypassing the encoding/decoding part if it doesn't have a use for it.

The encoder-decoder architecture consists of:
encoder:
C64-C128-C256-C512-C512-C512-C512-C512
decoder:
CD512-CD512-CD512-C512-C256-C128-C64

After the last layer in the decoder, a convolution is applied to map to the number of output channels (3 in general, except in colorization, where it is 2), followed by a Tanh function. As an exception to the above notation, BatchNorm is not applied to the first C64 layer in the encoder. All ReLUs in the encoder are leaky, with slope 0.2, while ReLUs in the decoder are not leaky.

The U-Net architecture is identical except with skip connections between each layer i in the encoder and layer n−i in the decoder, where n is the total number of layers. The skip connections concatenate activations from layer i to layer n − i. This changes the number of channels in the decoder:
U-Net decoder:
CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128[16].

3.2.4. Discriminator architectures(PatchGAN)

In order to model high-frequencies, it is sufficient to restrict our attention to the structure in local image patches. Therefore, we design a discriminator architecture – which we term a PatchGAN – that only penalizes structure at the scale of patches. This discriminator tries to classify if each N × N patch in an image is real or fake. We run this discriminator convolutationally across the image, averaging all responses to provide the ultimate output of D. The 70×70 PatchGAN discriminator architecture is: C64-C128-C256-C512.

After the last layer, a convolution is applied to map to a 1 dimensional output, followed by a Sigmoid function. As an exception to the above notation, BatchNorm is not applied to the first C64 layer. All ReLUs are leaky, with slope 0.2.

3.2.5. Hierarchical U-net structure

After we experimented with the previous U-Net structure, we found the generated image neglected details and did blur the cars a lot. So we applied a structure similar to 'Resnet', to concatenate different U-Nets together to form a generator. Here we call it Hierarchical U-Net.

As we have learnt previously, the Resnet structure could capture and amplify the local differences between different layers, and adjust the reality loss with these differences. By training for a long time, we could give out a precise image-classification solution. Inspired by this concept, we could also amplify the difference between different U-Nets, provided we already have had the two U-Nets training different properties of image separately. The training process then would pay more attention to the differences between low-level and high-level information, and finally achieve a fine-grained image in a relatively short time compared to the original U-Net.

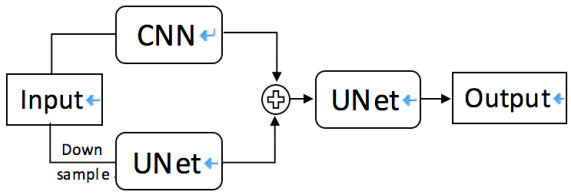


Figure 1: The flowchart to illustrate the Hierarchical U-Net and show the pipeline.

Suppose we first have one image sizing 256 x 256 x 3. First we downsample it and feed it into the first U-Net. Since a downsampled version contains a general grasp of the original image, the first U-Net could then be customized into a 'global trainer', training only the high-level information. Then we differentiate the original image and output image of the 'global trainer', and feed into the second U-Net. The second U-Net, seen as 'local trainer', would not only inherit output from 'global trainer', but could also train local details separately, without confusion with the 'global trainer'. That's why our Hierarchical U-Net would achieve a more precise result efficiently

3.2.6. Optimization

We alternate between one gradient descent step on D, then one step on G. As suggested in the original GAN paper, rather than training G to minimize $\log(1 - D(x, G(x, z))$, we instead train to maximize $\log D(x, G(x, z))$. In addition, we divide the objective by 2 while optimizing D, which slows down the rate at which D learns relative to G. We use minibatch SGD and apply the Adam solver, with learning rate 0.0002, and momentum parameters $\beta 1 = 0.5$, $\beta 2 = 0.999$.

3.2.7. Training details

Random jitter was applied by resizing the 256×256 input images to 286 × 286 and then randomly cropping back to size 256 × 256. All networks were trained from scratch. Weights were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02.

We use Cityscapes dataset, 2975 training images from the Cityscapes training set, trained for 200 epochs, with random jitter and mirroring. We used the Cityscapes validation set for testing. To compare the U-net against an encoder-decoder, we run both discriminator for 200 epochs, and we found out that the result for U-net is better than encoder-decoder, this is because we set batch size as 1 in the experiment and apply batch-norm on all layers of our network. And for batch size 1, it would zero the activations on the bottleneck layer. The U-net is able to skip over the bottleneck, but the encoder-decoder cannot, and that is why the encoder-decoder requires a batch size greater than 1. [14].

Furthermore, we test our generator networks by using different kernels, we change the kernel size from 4 to 2 to give the output more low-level details, however because lack of the global information, the output image is worse than the output image of kernel size 4. And we also change the last layer of discriminator to 16 × 16, so that we can get more global information to judge if the input image is real of fake, and according to judgment of metrics, the result it's still a little bit fall behind than our current architecture.

4. Metrics

Since we have implemented several different generator and discriminator structures, whose results turned out to be not bad observed with eyes, it is difficult to say which model has the best performance. Thus, it is important to put all the results under a certain metric to evaluate the quality of generated images, and then came to a conclusion which structure performs best.

The main idea of finding such metric is pretty obvious: the closer generated images fit label images, the better performance the corresponding generator has. Even though the basic idea is quite simple, it is not as easy as it seems to define such a metric after our careful reflection. Because it is so hard for us to simply quantify to what extent an image fits the label image, or we can say, target image.

The first thought that came into our mind is that, we can basically compare each pixel, and compute the difference between label image and generated image and sum them up. This 'difference' can be defined as the absolute value of difference or other kinds of difference, there are many functions available, but in other words, it is a pixel-wise metric. This metric is the most intuitive and will be easiest for us to understand and use. However, it has a fatal flaw, that is, it overemphasizes the significance of regional difference and ignores the overall characteristics. For example, we implement two different generators to generate a building image using the same label image, we cannot simply say image A has a better quality than image B only due to its color is more similar with the label image. Because the color of the building, whatever black or white, does not influence the quality of the output image. But if we put these two images with different colors under pixel-wise metric, the results will be much different, since pixel with different color has different pixel value. Thus, a image with higher scores in pixel-wise metric does not necessarily mean it has a better performance, which will mislead our judgment. Then we tried to figure out a solution to countermeasure this problem, such as do local averaging or convolution to extract as more features as possible. We even thought of training another neural network especially for evaluating generated images, but still, we cannot train a network without labeled images. In other words, pixel-wise metric is easy to understand and implement, though, it has some flaws and it may judge incorrectly under some circumstances.

When our thinking goes deeper, we believe perhaps there is still no perfect or mature evaluation protocol to quantitatively evaluate all conditional GAN cases. Because this kind of evaluation algorithm can be considered as a sophisticated discriminator, if we could construct this algorithm (discriminator) to perfectly evaluate the quality of generated images, then we would get the generator to generate perfect images at the same time, and our problem is solved. But obviously it will not happen. So it does not matter whether the metric is pixel-wise or not, all metrics for sure have limitations, but for each metric we can still get the correct ranking in most cases, and we can implement several different metrics to reduce these limitations and get a more persuasive results.

Since we cannot have a perfect evaluation protocol, we then tried to introduce 'human' as a discriminator to get a more accurate evaluation. We conducted a manual evaluation based on a questionnaire to let volunteers quantitatively decide the quality of a series of generated images. We assumed that they all did not have any idea about the correspondence between images and which model they are generated from, and they have to score each image from 1 to 10. Finally we collected all the scores from different volunteers and get the average score of each model, and if the number of volunteers is big enough, we have the confidence to say that the model with the highest score has the best performance.

Meanwhile, in order to get rid of human's subjective influence to achieve a more objective evaluation, we still implemented the pixel-wise metric. The results are as follow.

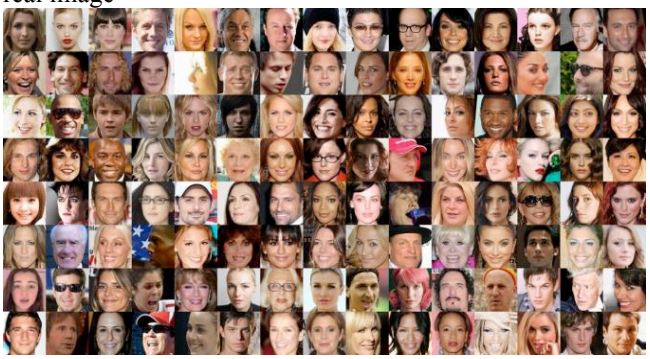
5. Results

5.1. PG-GAN result

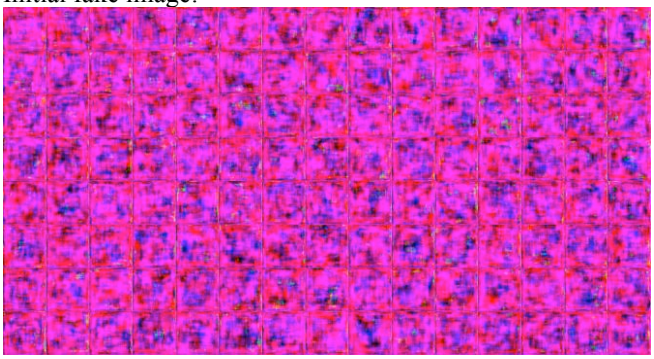
5.1.1. CelabA dataset

CelabA fake images generation CelabA Datasets link: (http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

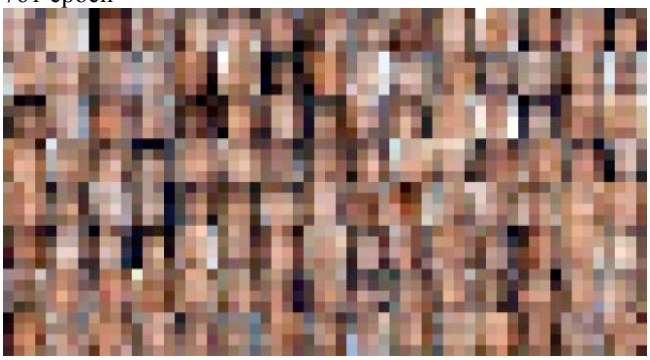
real image



Initial fake image:

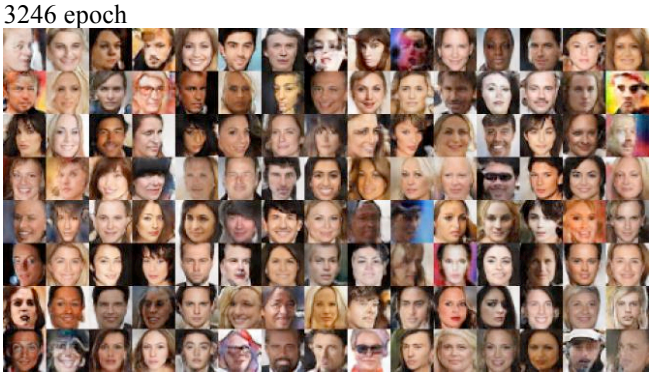


761 epoch



1603 epoch

3246 epoch

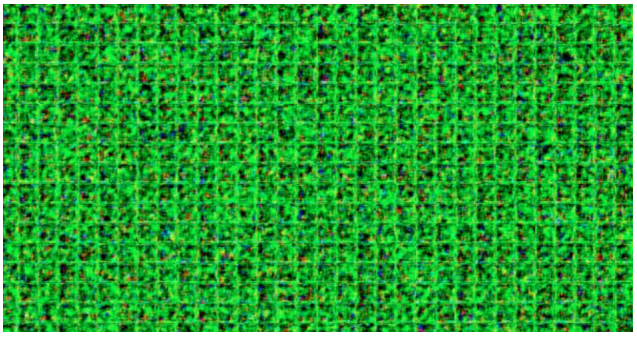


5107 epoch

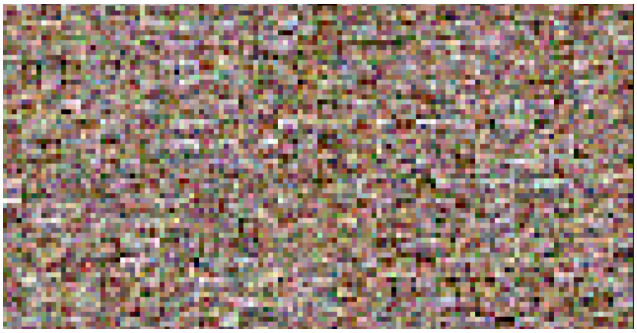
5.1.2. CIFAR-10 dataset

The CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html)

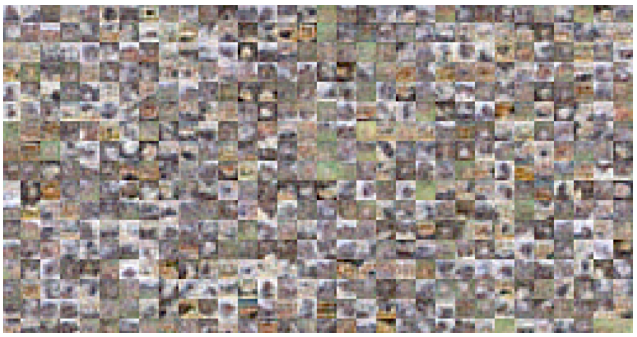
Initial random images (000000 ticks)



Generated images 1 (000160 ticks)



Generated images 2 (001182 ticks)



Generated images 4 (002705 ticks)

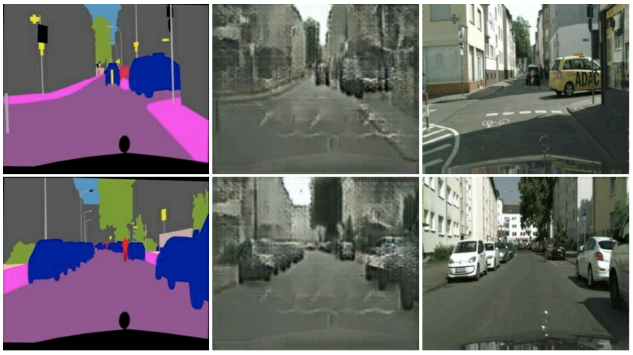
Generated images 4 (003547 ticks)

Generated images 4 (004147 ticks)

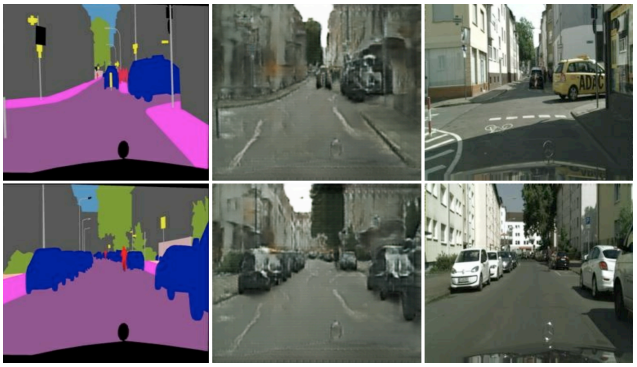
5.2 Pix2pix translation result

Cityscapes dataset link:
(https://www.cityscapes-dataset.com/)

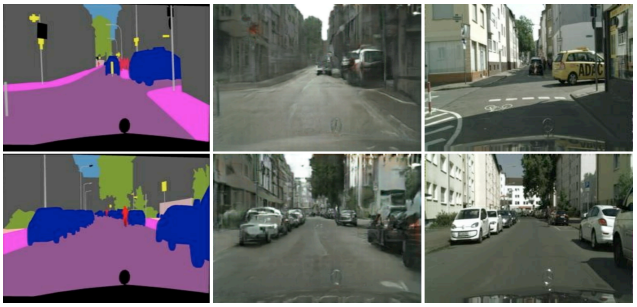
3 epoch



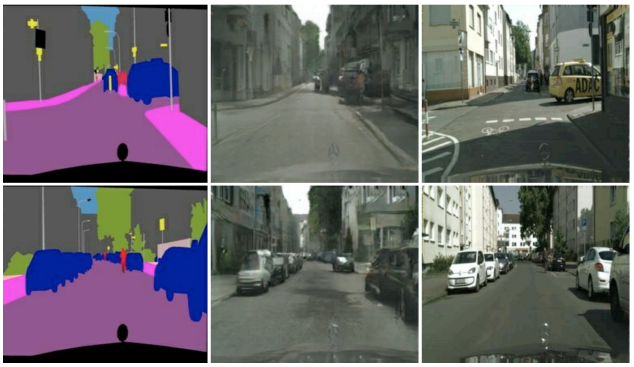
10 epoch



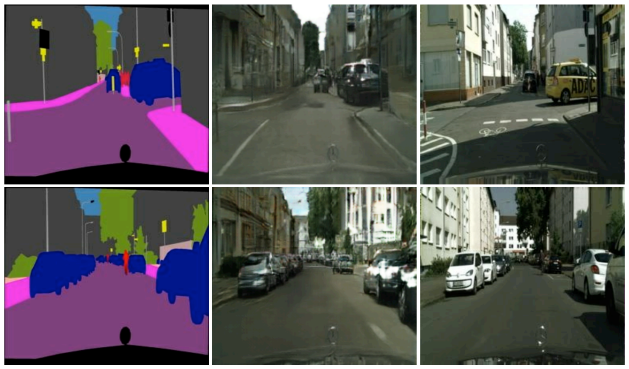
100 epoch



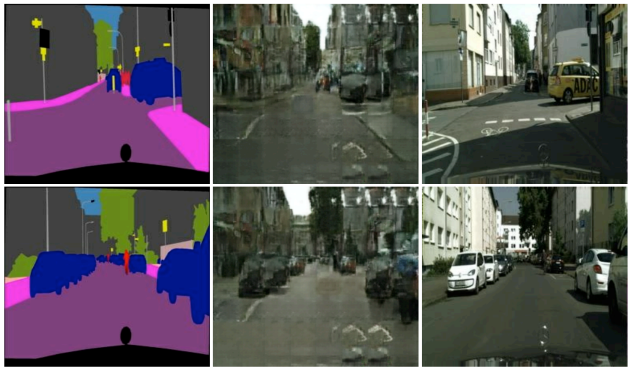
200 epoch



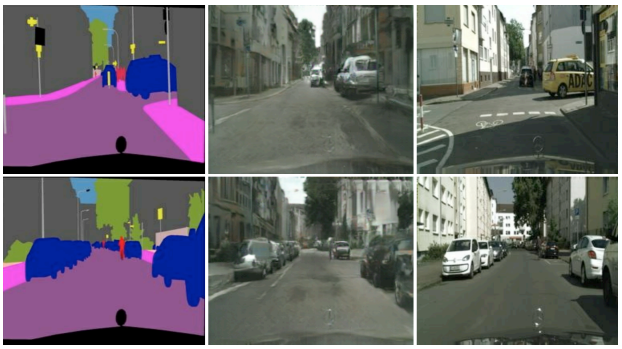
Hierarchical U-net



Kernel = 2



Discriminator output =16 × 16



6. Future works

Due to the limited time and capabilities, we have not made major changes to the main ideas, GAN structure, training methods of the two experiments in this project: progressive GANs and pix2pix translation. However, we have made some reasonable and new attempts on the network structure of U-Net, as well as some in-depth thinking on metrics, and we got some conclusions under these metrics.

In the course of the experiment, we continued to deepen our understanding of the project and also had some thoughts on future work of progressive GANs and image to image translation. For progressive GANs, the network introduces a new training mechanism to dynamically add layers, which allows the network growing progressively from simple to sophisticated. We believe that this mechanism will be introduced in more and more other GAN networks in the future.

For pix2pix translation, we saw the huge potential in this field. It does not translate image to text like humans describing a painting in English, instead, it translates directly from image to image, which is more efficient than human translation.

It does not use text to translate pictures, instead, it to specify the "style" of the pictures to generate pictures corresponding to "style". By reviewing some related articles, we realize that this kind of translation is actually reversible, and some networks have already achieved reversible translation such as Cycle GAN. For example, we

can easily translate a cat to a dog by implementing Cycle GAN, and we can do the same in reverse. From a larger perspective, we can easily replace any objects with another, and replace one singer's face to Michael Jackson in a MV, which will perhaps profoundly change many industries in the future.

References

[1] Tero Karras,Timo Aila,Samuli Laine,Jaakko Lehtinen,Progressive Growing of GANs for Improved Quality, Stability, and Variation,2018

[2] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 09 2016.

[3] Kingma and Welling, Auto-Encoding Variational Bayes ,2014

[4] Goodfellow, I. NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv'16.

[5] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. "Improved Techniques for Training GANs". arXiv preprint 2016.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville. "Improved Training of Wasserstein GANs". arXiv preprint 2017.

[7] Berthelot et al., "BEGAN: Boundary Equilibrium Generative Adversarial Networks". arXiv preprint 2017

[8] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". arXiv preprint 2016

[9] I Durugkar,I Gemp,S Mahadevan,Generative Multi-Adversarial Networks, 2016

[10] A Ghosh,V Kulharia,V Namboodiri,PHS Torr,PK Dokania Multi-Agent Diverse Generative Adversarial Networks ,2018

[11] KE Clark,H Lopez,BA Abdi,SG Guerra,X Shiwen,Multiplex cytokine analysis of dermal interstitial blister fluid defines local disease mechanisms in systemic sclerosis ,2015

[12] Y Kao,K Huang,S Maybank,Hierarchical aesthetic quality assessment using deep convolutional neural networks ,2016

[13] G Zhang,E Tu,D Cui,Stable and improved generative adversarial nets (GANS): A constructive survey, 2017

[14] Phillip Isola,Jun-Yan Zhu,Tinghui Zhou,Alexei A. Efros,Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014

[16] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[17] https://github.com/RemexHu/PGGAN-Project

[18] https://github.com/RemexHu/Pix2Pix