The Elements of Style Transfer

Alexander Chen
Department of Mathematics
alexanderKchen [at] princeton [dot] edu

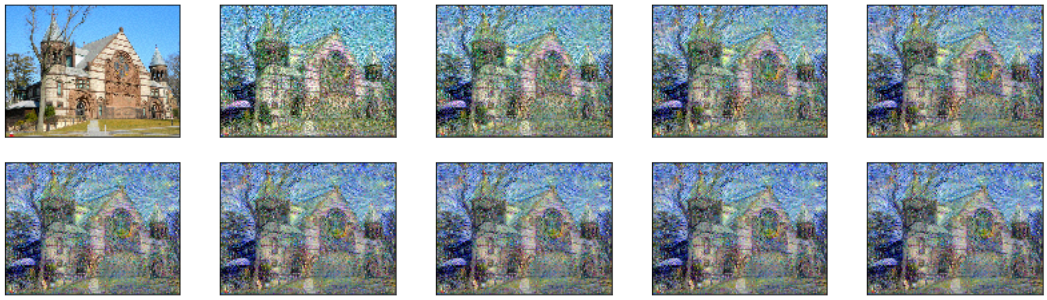


Figure 1: Van Gogh's starry night style applied to Alexander Hall, Princeton University

Abstract

Neural artistic style transfer (NST) for images found its first success by using deep learning and convolutional nets. A number of unsupervised NST learning methods are surveyed here including GANs and Gatys' original method.

1 Introduction

The fundamental problem of style transfer is to extract and apply the 'style' of artpiece S, say a painting, music recording, or written work, to another artpiece X to generate a new artpiece Y that matches the content of X but with the style of S. The initial difficulty is how to define the 'style' of a picture or piece of literature, but in recent years deep neural networks have provided a way.

In 2015, Gatys et al introduced a method for deep NST.[1] It is based on using a pre-trained convolutional net (CNN). An input image is fed into the net, and then deep renormalized layers are used as a representation of the perceived 'content' of an image. Correlations between these features represent the 'style' of the image. The goal of the training was to reduce the 'style distances' plus the original 'content distances' between images.

Figure 2: From Gatys et al, Figure 2ab [1]

Applications have since been found in photorealistic style transfer and image synthesis.[2]

The year before in 2014, Goodfellow et al introduced generative adversarial models (GANs) as a means of generating stuff to emulate something as close as possible, broadly stated.[3] For example let's say we want to draw a painting in the style of Monet or Van Gogh.[10]

GANs have also been used for face morphing (of the Mission Impossible face mask kind), face-swapping, and even 'anime' (cartoon) design.

https://twitter.com/_Ryobot/status/1095160640241651712

https://twitter.com/gwern/status/1095131651246575616

Most recently a paper by NVIDIA introduced a style-GAN.[12][13][14] It creates new faces such as celebrity-styled faces.

https://thispersondoesnotexist.com/

In contrast to Gatys, which uses a pre-trained network to train the output pistache, these GAN methods train the networks to create the output pistache. A question we ask is how else further these methods compare.

1.1 Description of Data

The images are taken from wikipedia under the wikimedia license.[11]

- https://upload.wikimedia.org/wikipedia/commons/3/3e/Barbara_Engelhardt_Princeton_2019.jpg (picture of Barbara Engelhardt, taken by the author with her permission, and uploaded to wikipedia)
- https://upload.wikimedia.org/wikipedia/commons/b/bd/Italian_Renaissance_Princeton%2C_NJ.JPG
- https://upload.wikimedia.org/wikipedia/commons/d/d7/Green_Sea_Turtle_grazing_seagrass.jpg
- https://upload.wikimedia.org/wikipedia/commons/0/0a/The_Great_Wave_off_Kanagawa.jpg
- https://upload.wikimedia.org/wikipedia/commons/b/b4/Vassily_Kandinsky%2C_1913_-_Composition_7.jpg
- https://upload.wikimedia.org/wikipedia/commons/0/00/Tuebingen_Neckarfront.jpg
- https://upload.wikimedia.org/wikipedia/commons/thumb/e/ea/Van_Gogh_-_Starry_Night_-_Google_Art_Project.jpg/1024px-Van_Gogh_-_Starry_Night_-_Google_Art_Project.jpg

For style-GAN, we use the Flickr Dataset (FFHQ). This includes up to "70,000 high-quality PNG images of human faces at 1024x1024 resolution (aligned and cropped)." Note: we do not use the entire dataset (see Discussion section for more). [12]

https://github.com/NVlabs/stylegan

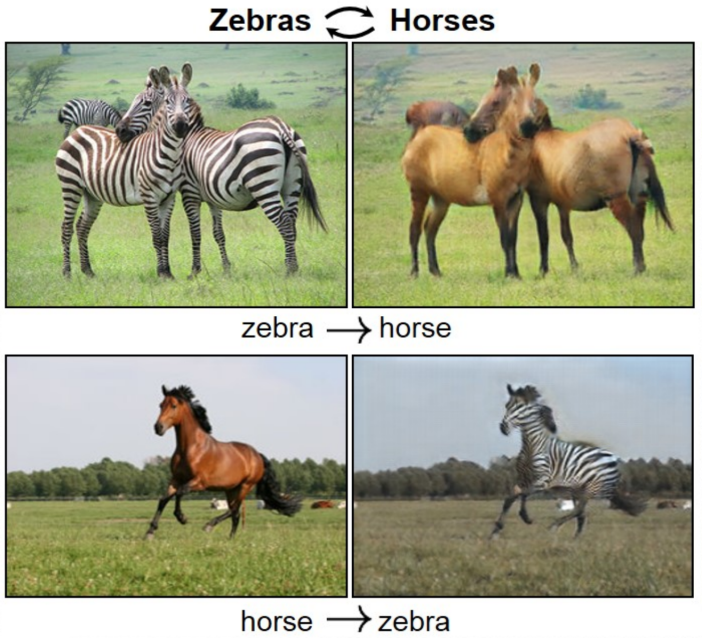
How much processing is necessary? Not much at all. This is because Gatys' method and GAN only require one instance of an image to deduce style.

We could try to aggregate multiple images to deduce an *average* style: for example by using say Picasso or Frida Kahlo's entire portfolio to determine their average style. But that would be more expensive than necessary for the scope of this paper, so we stick to using one photo to determine style because it currently works well enough.

2 Related work

Cycle GAN has been used before for style transfer as well. It has been used to zebras into horses and horses into zebras. This is used for image to image transformation, among other applications. [10] The code and data are from:

`https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`



Johnson, Alahi, and Li Fei Fei extended Gatys' model by introducing an L2 regularized total variation metric. Moreover rather than measuring the distance between the content pictures, they measured a 'perceptual' distance between higher layer representations of the pictures. Their methods sped up Gatys' original model three-fold.[4]

There have also been ideas of applying GAN to music. Say you want to turn the song 'All my loving' into a 'jazzier' or 'funkier' version, in the style of Stevie Wonder. This is a slightly different problem from the one solved by Ji-Sung Kim's *deepjazz*, which uses LSTM RNNs to improvise music from scratch (rather than applying different music genre styles to existing scores).[5] Music GANs provide a way.

3 Methods for deep style

Two main methods: the one used in Gatys' paper (which we refer to as RST below) and GANs.

Tensorflow was used with tf.keras and Adam optimizer for gradient descent.[11][15] Code was based on that written by Yuan.[11] Google Colab was used for cloud computing and GPUs. In the future we may consider using our own TPUs and NVIDIA chips, programming in CUDA even.

3.1 Evaluation Metrics

The evaluation metrics are the loss functions L described in the following sections.

3.2 Renormalized Style Transfer (RST)

Some authors (see [9]) refer to the method proposed by Gatys as Neural Style. However this is a bit broad as style-GAN would also classify as an NST algorithm. So in this paper we refer to Gatys' method renormalized style transfer (RST), based on *renormalization*, an idea originally from physics

that involves coarse-graining over detailed data to extract more general feature representations.[6] One could try Gatys artistic transfer (GAT) but two others co-authored the paper too.[3]

As described in Gatys et al, [1] we have a style image S and a content image C. We want to generate x (or 'pastiche') that has the content of C but the style of S. The ij pixel is represented by $C(ij)$. Suppose our neural network is N. The original paper uses the VGG-19 CNN (a 19 layer convolutional net used with near-human performance in object recognition).

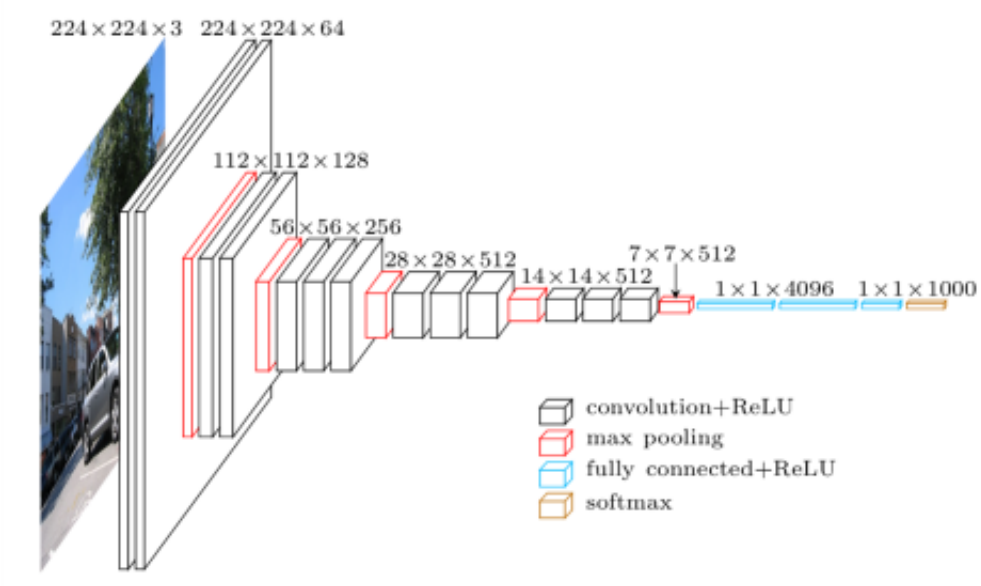


Figure 3: Pre-trained VGG-19 CNN, a 19 layer convnet used for RST[1]

Suppose we can partition N into feedforward layers (so that two consecutive layers form a directed bipartite graph, not necessarily complete / fully connected). Each layer consists of weights. Let $x_l = N_l(x)$ be the lth layer when image x is fed into the neural network (also known as the 'encoding' of x in layer l). The 'content metric' at layer l is defined as

$$L_c^l(C, x) = |C_l - x_l|^2$$

where the RHS is the pixelwise mean-square difference.

The style metric is defined as

$$L_s^l(S, x) = |x_l x_l^T - S_l S_l^T|^2$$

where x^T denotes the transpose so xx^T is the correlation matrix (also called Gram matrix).

In our code we take the average over all M layers:

$$L_s = \frac{1}{M} \sum_l L_s^l.$$

The total content loss L_c is defined similarly.

For RST we want to minimize a weighted sum of both the style and content losses with respect to x:

$$L(C, S, x) = L_c(C, x) + kL_s(S, x)$$

where k is a hyperparameter that controls how much style we want to apply.

3.2.1 Training

Originally x is random (say gaussian-generated). We apply the following gradient descent algorithm [15] to modify the pastiche x:

$$\Delta x = -\eta \partial_x L$$

Here Δ denotes the iterative change, and η tunes the learning rate.

That's it. We don't modify any of the weights of the network N. We repeat the above until convergence or cutoff.

3.3 GANs

To understand style GANs we first introduce what a GAN is.

As outlined in the Goodfellow paper, [3] in GAN we have two neural networks, a discriminator $D(x')$ that determines whether x' is real or fake, and a generator $G(z)$ that creates a new image from z. Usually the z is generated from random noise (e.g. Gaussian).

Suppose x are the real images we want to imitate. Ideally we want $D(x) = 1$ all the time. Suppose $x' = G(z)$ is a fake image generated by G. Ideally we want $D(x') = 0$ all the time. Hence we want to maximize the probability

$$\begin{aligned} p &= \prod_x D(x) \prod_z (1 - D(G(z))) \\ L(D) = \log p &= \sum_x \log D(x) + \sum_z \log(1 - D(G(z))) \\ &= E_x \log D(x) + E_z \log(1 - D(G(z))). \end{aligned}$$

which is equivalent to maximizing the log likelihood functional in the second line. The notation E_y denotes expectations over the random variable y.

At the same time the generator wants to make images as realistic as possible so it tries to minimize the maximum of the above. Thus for GANs our goal is to solve the following minimax game:

$$\min_G \max_D L(D, G) = E_x \log D(x) + E_z \log(1 - D(G(z))) \tag{1}$$

or at least find approximate solutions D_u, G_v which as neural nets are typically parameterized by neuron weights u, v.

The above also shows that GAN has a natural game-theoretic formulation (actor-critic model). If we can specify the probability measures of $x \sim \rho_r$ and $z \sim \rho_g$ exactly, then Goodfellow shows that the optimal solution is $\rho_r = \rho_g$.[3] In real applications we do not necessarily know what ρ_r is.

3.3.1 Gradient descent

To train the model we apply m-minibatch gradient descent [15] in the following way (as described in [3]):

1. sample m random inputs $z_1, \ldots, z_m \sim \rho_g$ e.g. usually use a gaussian $N(0, 1)$
2. sample m real images $x_1, \ldots, x_m$ from the training set
3. update D_u via ascension

$$\Delta u/\eta = +\partial_u L = \partial_u E_x \log D_u(x) = \partial_u \frac{1}{m} \sum_i \log D_u(x_i)$$

4. then update G_v via descension

$$\Delta v/\eta = -\partial_v L = -\partial_v \frac{1}{m} \sum_i \log(1 - D_u(G_v(z_i)))$$

5. repeat for any finite number of epochs

3.3.2 Cycle GANs

We can directly apply the above framework to NST as described in [9]. Let x be the desired style image and z is the content image. We want to train G, D as above so that the produced image $G(z)$ matches the style of x with the content of z.

This time we have four networks: G_x generates x from z; D_x determines whether x is genuine or not; and G_z, D_z serve similar functions in reverse. Note the subscript does not denote parameterization but that G_x tries to generate a clone of x given input z. Take equation (1) and define

$$L(D_x, G_x) = E_x \log D_x(x) + E_z \log(1 - D_x(G_x(z)))$$
$$L(D_z, G_z) = E_z \log D_z(z) + E_x \log(1 - D_z(G_z(z)))$$
$$L_c(z, x) = E_x |G_x(G_z(x)) - x| + E_z |G_z(G_x(z)) - z|$$

Ideally we want G_x and G_z are inverses so we also try to minimize the 'cycle consistency loss' in the third line. We use L1 norm (though L2 could also work).[9] The point of this is to make sure that the content of the original image z is still preserved but that G_x is still able to apply style features to make z feel like x. The total action we want to minimize with respect the two pairs of D, G is

$$L = L(D_x, G_x) + L(D_z, G_z) + L_c(z, x).$$

3.3.3 Style GANs

The basic architecture of style-gans is the progans described in [13]. Essentially progan is a GAN described above but as the training progresses it adds more layers to the generator network G. Initially it starts with one shallow layer to produce low resolution images (4×4). After a few iterations of learning 'coarse' features of an image, the G adds another layer to produce slightly higher resolution images (8×8). It continues doing this, learning finer more detailed features as training progresses until convergence. The style-gan described by NVIDIA labs uses this fundamental idea but adds a lot of other statistical tricks to improve learning, diversity in image output, and 'stability' (described in discussion).[14]

4 Results

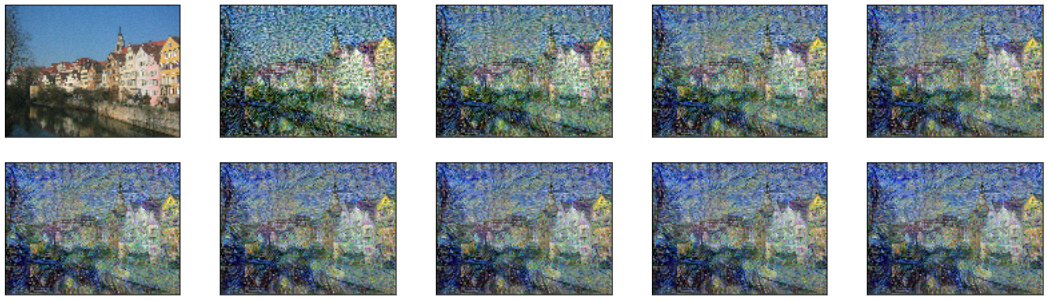
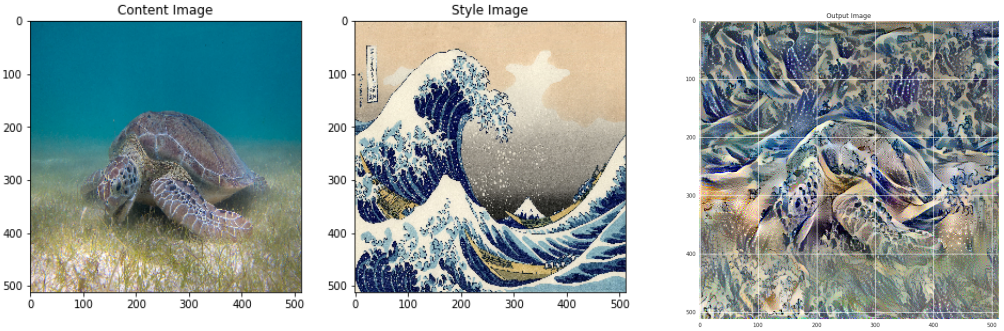


Figure 4: Tuebingen + Starry Night

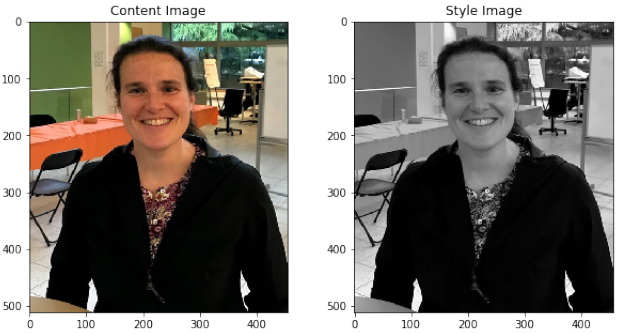


(a) C and S. Turtle + Great Wave.

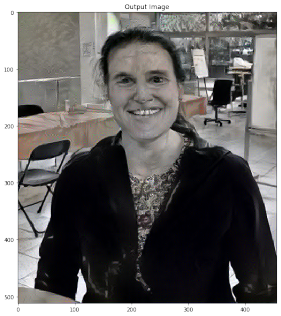
(b) Result of RST

Left is the original image of a turtle. Middle is the artistic style we want to apply.

Figure 6: Style GAN faces. These people (probably) don't exist. [14]



(a) Left is original. Right is original with grayscale.



(b) Result of RST / CGAN: almost matches.

Note that if we use the left original image of Barbara as the style image as well (so that $S = C$ are identical) then we get images that are very similar to the original (but in noise), but in the end the best image looks exactly like the original. The difference is subtly detectable to the naked eye and is not included here to save space but one will find it in the colab result html.

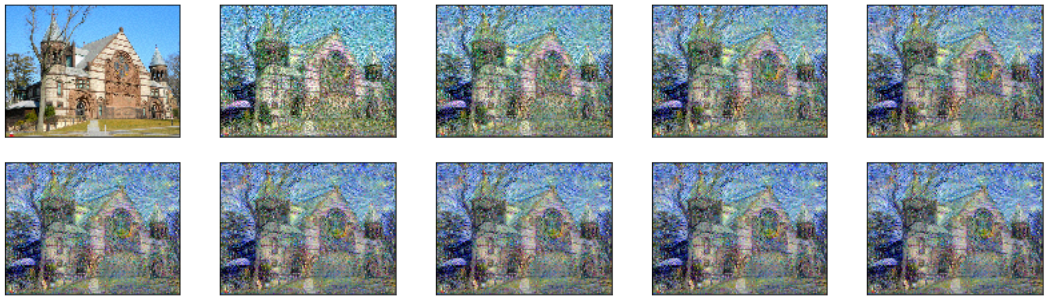


Figure 8: Alexander Hall + Starry Night: 1000 iterations 10 snapshots. Both the content and stylistic losses were found to decrease over successive iterations. However for other tests this was not always the case.

Many more results are found in the colab result html and online in the art gallery at:

`https://drive.google.com/open?id=1uoSOo3W3OHwBwbQHrKsIzIe6bjZR5eWo`

5 Discussion and conclusion

Training takes a few minutes for each style-content image pair.

If we view RST as a function $f(C, S)$ mapping the content image and style image to produce a new image, then we would want $f(C, C) = C$. We applied this to Barbara's image and indeed got an output image perceptually similar to the original. Now suppose we apply grayscale to the image

so that $S = g(C)$. Note S has the same 'content' as C except now it is gray. We would want $f(C, g(C)) = g(C)$. We could apply other filters, inversions, etc. and we would want our style transfer algorithm to learn these (generally nonlinear) mathematical operations as well. Textural features seem to be transferred but this shows that 'style' has a pretty broad definition that is not so easily captured in one stroke. A better theoretical understanding of the mathematical properties of such a transformation would be desirable.

A practical and commercial goal would be to reduce uncanny valley as much as possible and create as realistic pictures as possible. One can try to put a number on the 'content difference' or the 'style distance' as we have done above for RST. But ultimately to quantify 'realism' for now mostly lies in the eyes of the beholder; and current research on NST seems to focus on accelerating real-time generation methods and creating even more persuasive style transfers and imitations.

The two methods mentioned in the paper RST and GAN may be equivalent or dual. Sometime in the future a mapping between the models may be proposed. For example above say $C = x$ is the desired content image. z is the random input. RST seeks to update z given fixed network N. GAN seeks to update the networks D, G so $G(z)$ is the output image.

We could also try using the *entire* FFHQ dataset, but we would need more time and resources. According to NVIDIA estimates, even with GPUs (Tesla V100) training with the *entire* FFHQ can take up to five weeks, way beyond the scope of this project. [12] It would help to understand better and maybe even simplify the reigning style-GAN model, which itself is a great technical engineering feat.[14]

There are some other current theoretical problems showing that GAN does not always converge to a stable equilibrium (so once every while it might produce a very bizarre image, sometimes referred to as 'mode collapse'). Eventually we want a way to refine the existing GAN so that we can guarantee it always produces 'good' images with reasonable convergence (in a reasonable amount of time).

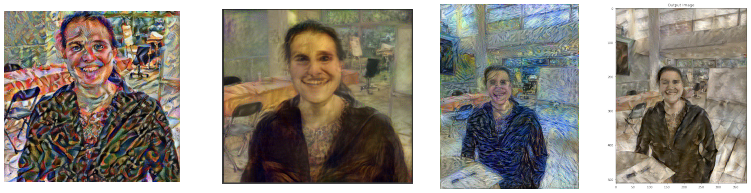
Various experimental measures have been taken to ameliorate this:[7] for example by replacing the cross entropy loss in equation (1) with the Wasserstein metric (or Earth mover's distance). The resulting model is called WGAN and remains an area of current research.[8]

Will AI and deep style replace artists? We hope not because nothing quite beats the human touch.

6 Acknowledgements

This was written for COS424, a Princeton University class taught by Barbara Engelhardt. With her permission a picture of her was taken by the author and numerous styles were applied to her liking.

(a) Original cropped (b) Barbara Monroe (c) Line / contour (d) Bee of Troy (e) Bee Wave



(f) Bee Vassily (g) Mona Beesa (h) Starry Bee (i) Beecasso

Figure 9: Barbara Engelhardt, Photo by Alex Chen

References

[1] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. *A Neural Algorithm of Artistic Style* https://arxiv.org/pdf/1508.06576.pdf

[2] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. *Image Style Transfer Using Convolutional Neural Networks.* https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, et al. *Generative Adversarial Networks* https://arxiv.org/abs/1406.2661

[4] Justin Johnson, Alexandre Alahi, Li Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution* https://arxiv.org/pdf/1603.08155.pdf

[5] Ji-Sung Kim. *deepjazz.* https://deepjazz.io/

[6] Kenneth G. Wilson. *The renormalization group: Critical phenomena and the Kondo problem* https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.47.773

[7] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. *Improved Techniques for Training GANs* https://arxiv.org/pdf/1606.03498.pdf

[8] Martin Arjovsky, Soumith Chintala, and Leon Bottou. *Wasserstein GAN.* https://arxiv.org/pdf/1701.07875.pdf

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. UC Berkeley. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* https://junyanz.github.io/CycleGAN/

[10] Jonathan Hui. *Some cool applications of GANs.* https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900

[11] Raymond Yuan. *Neural Style Transfer.* https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398

[12] Mos Zhang, Michael Sarazen. *NVIDIA Open-Sources Hyper-Realistic Face Generator StyleGAN.* https://medium.com/syncedreview/nvidia-open-sources-hyper-realistic-face-generator-stylegan-f346e1a73826

[13] Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen. NVIDIA. *Progressive growing of GANs for improved quality, stability, and variation.* https://arxiv.org/pdf/1710.10196.pdf

[14] Tero Karras, Samuli Laine, Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks* https://arxiv.org/pdf/1812.04948.pdf

[15] Li Fei Fei. CS231n Convolutional Networks for Visual Recognition (Stanford). *Optimization: Stochastic Gradient Descent,* http://cs231n.github.io/optimization-1/