AI Artist with Reinforcement Learning and Graph Neural Network

Keliang Luo [1, †, *], Xinyi Wang [2, †], Qingfeng Zou [3, †]

[1] School of Science and Technology, Hong Kong Metropolitan University, Hong Kong, China

[2] School of Big Data and Software, Chongqing University, Chongqing, China

[3] College of Letters and Science, University of California, Los Angeles, USA

* Corresponding author email: s1217945@hkmu.edu.hk

†All authors contributed equally

Abstract. Recently, researchers have had much progress in the field of AI painting, inventing advanced machine learning models like Stable Diffusion. However, past methods mostly neglect the clients' customized requests and need for humanized service. Facing this weakness, this paper constructs an AI to receive text input and to literally paint the image in a humanized way with a customized style. It analyzes the clients' style requests and breaks down the need into two parts, the painting style, and the painting technique. After generating the initial image with Stable Diffusion models, this AI pre-process it with graph neural networks to satisfy the need for customized painting styles. Then, Reinforcement Learning is implemented to train the AI to paint the image like human painters with distinct strokes. In this step, the strokes can be modified to mimic various painting techniques like oil painting, pencil, etc. By two aspects of the style, this AI artist provides clients with a personalized painting with and a video of the full painting process.

Keywords: AI Paint; Reinforcement Learning; Ai Artist.

1. Introduction

Since the beginning of artificial intelligence, natural language processing and visual art have been recognized as two intricate fields due to their close relationship with human cognition and wide applicability [1]. For a long time, people defied the possibility that AI could process creativity [2]. However, after decades of hard work and the advancement of hardware, researchers have made impressive progress in these fields. Through advanced machine learning models like Stable Diffusion, which could intake descriptions in the form of simple text and produce astounding visual art with gorgeous details. A painting produced by AI in a few minutes could take hours for a human painter to accomplish, not to mention its advantage in learning speed compared with humans [3].

However, even algorithms like stable diffusion have some drawbacks, especially for clients without programming experience. On the one hand, it is extremely convenient with pre-trained models: text in and images out. On the other hand, it is hard for the client to pick a certain painting style or certain painting technique they desire for their painting unless they have pre-trained specific models. They also get nothing else besides the painting, just like a product from the factory assembly line.

Facing these drawbacks, this paper would create an AI that is more human with the spirit of service to their clients, just like a professional street artist who provides painting services to the public. This AI can understand the client's demands from a plain word description, allowing them to decide their preferred painting style or painting technique. Best of all, it paints out the painting in front of the clients rather than sending it to their mailboxes. That means our AI will produce a video of the entire painting process, just like a human painter with their own customized style.

This paper accomplishes such a goal with a combination of three main models and three steps. First, it uses a stable diffusion model to generate an initial image with the clients' input text description. Then, facing the clients' requirement for a specific style, it could pre-process this initial image with a graph neural network that's built on customized image sets. It modifies the initial image

to a specific style chosen by the client, which could be the style of a famous painter or a certain art school. Last, to teach the AI to paint in a humanized way with required painting techniques, this paper implements a reinforcement learning model with a few strokes literally painting on a flat. This algorithm employs these strokes as agents, and the reward is how close the current picture is to the image from the previous step. In this way, it can also handle the requirement of various painting techniques by modifying the parameters of the strokes or migrating each stroke to a specific style, provided by OpenCV, like oil painting or color pencil. Finally, the AI paints the picture like a human painter, and the painting process is visible to the clients like we present in Figure 1.



Fig 1. The left-top image is the initial image from Stable Diffusion. The Next three images are taken from the video from the painting process, with an oil painting style transformation.

2. Related Work

2.1 AI Painting

One of the most significant pioneers in AI painting is Harold Cohen, who developed AARON in 1973. He was determined to create an algorithm that can consistently produce evocative paintings, and applied that ideas on AARON, a painting machine that can automatically paint in an abstract style [4]. However, the core of his project was still a set of rules dedicated directly by the artists, which is hard to be considered as artificial intelligence [5]. Regardless of this defect, the advance performance of AARON draws lots of public attention into this field at that easily age, leading to the later progress.

In the next few decades, there has been an exponential increase in the number of digital art collections online, which provides the foundation for the study in AI painting [6]. Until today, projects in AI paintings rely heavily on the online database with large quantity of images, especially those with clear labels and strict classifications [7]. With this development, Ian Goodfellow and his colleagues introduced the generative adversarial networks (GAN) in 2014 [8]. Comparing with its predecessor, it utlizes machine learning technology to learn from the large online image datasets and be able to generate images automatically. It has two parts of networks, the generator and the discriminator, who keep competing with each other and improve themselves in the process [8]. Besides its effectiveness in learning the aesthetics from human painting, GAN also has remarkable

achievements in image segmentation, which leads to its wide application in face detection, traffic controlling and even medical fields [9].

In 2022, the emerge of Stable Diffusion has brought unprecedented public attention to the field of AI painting because its excellent performance in mimicking human painters. Based on latent diffusion models, it can produce images with text input by iterating through a space of latent representations to "denoise" the input and then decoding the representations into complete images [3]. Due to its popularity, there has been some improvement to the algorithm, like Karras's work in reducing the time cost of the learning process [10]. Overall, it is simple and effective for the users with pre-trained models: input the text description and output the image.

2.2 Image Style Transformation

In the beginning, researchers studied the image style transformation mainly based on mathematical models with statistics information of the images. They manually collected and calculated the statistical features of the image texture and apply to other images [11]. This process requires lots of calculations and complex steps, thus cost heavily in time and efforts.

With the development of computer vision and deep learning, researchers started to use convolutional neural network to capture features from the images. In 2014, Simonyan and Zisserman invented VGG Net, which can learn to classify unseen items with high accuracy by extracting complex features from images [12]. At these points, although the features have been extracted, it is too complex to be understood or utilized by researchers to perform style transformation. Finally, at 2015, Gatys and his colleagues process the features captured by VGG 119 with Gram matrix to get the texture information of the image [13]. Later, Gatys improved his methods and constructed the graph neural networks that can record the images' RGB value in mathematical form which can be apply to other images to achieve style transformation. This method can produce ideal and intuitive result in imitating the painting style of some famous artists [14].

Besides capturing the features from images, there are also many other ways to transform the style of an image. This paper was also inspired by the study of Huang's team, which train the AI to paint an existing image in a humanized way. Their reinforcement learning algorithm uses strokes as agents, trying to paint as close to the original painting as possible [15-16]. Through experiment, we realized that by distinctly changing the features of the stroke agents, we can paint the same images with various textures, similar to different painting methods. That means we can build on this algorithm to fundamentally change the painting style during the painting process, like changing the AI artist's painting techniques.

3. Methodology

There are three sections to our endeavor as a whole. "Use stable diffusion to create images (text to picture)", "style change (change through value iteration)," and "AI painting (using pre-trained stroke and renderer, let AI draw the matching picture)" are the three.

3.1 The Process of Stable Diffusion Generate Images

Stable diffusion is founded on a form of diffusion model known as latent diffusion. The auto-encoder (VAE), a U-Net, and a text encoder are the three major components of latent diffusion.

Those that desire to employ stable diffusion produce images through text descriptions, according to the general structure of Latent Diffusion Models which is presented in Figure 2. First, we must train an AutoEncoder, which includes encoder(ε) and decoder(D). The encoder can then compress the image, followed by the diffusion process in Latent Space, and lastly the decoder to restore the original pixel space.

To speed up picture production, a spreading procedure called "Departure to Latent Space" will be conducted on the compressed image. The forward diffusion process is performed on the latent compressor, as indicated in the image, and the noise predictor is really trained to forecast the noise in

the compressed representation (latent space). We create the data for training the noise predictor using the forward procedure (encoder with an autoencoder). After training, we may create pictures by reversing the process (using the autoencoder's decoder).

Conditioning Mechanisms is located on the far-right side of Figure 2. This encompasses both unconditional and conditional image creation. It primarily carries out these activities by acquiring a conditional denoising autoencoder. So, using a conditional denoising autoencoder, we can regulate the picture synthesis process (τ_θ).

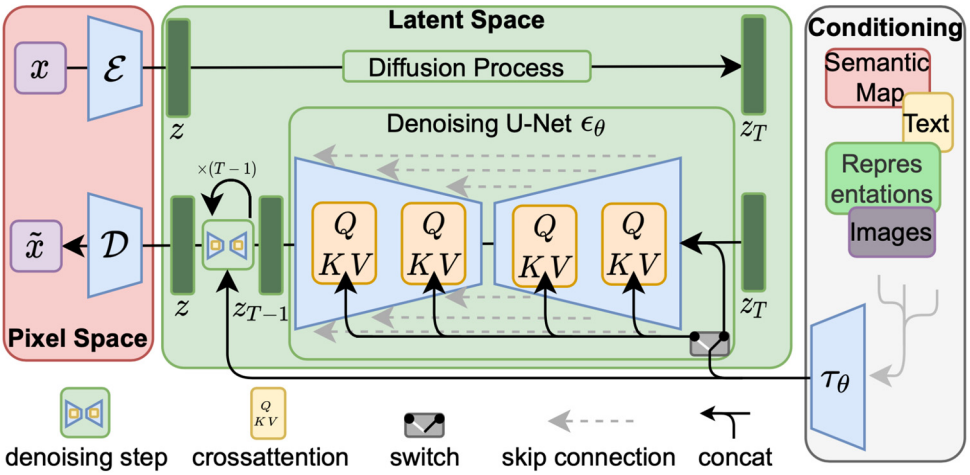


Fig 2. Latent Diffusion Models' overall frame diagram [17]

3.2 Pre-process the Image with Graph Neural Network

In order to transfer the styles from one picture to another, neural style transfer (NST) employs a pretrained convolutional neural network (CNN). This is accomplished by establishing a loss function that makes every effort to keep the disparity between the produced picture, the style image, and the content image to a minimum.

An already-trained convolution neural network is used for neural style transfer. Then, NST provides the following inputs to establish a loss function that smoothly merges two photos to produce aesthetically pleasing art.

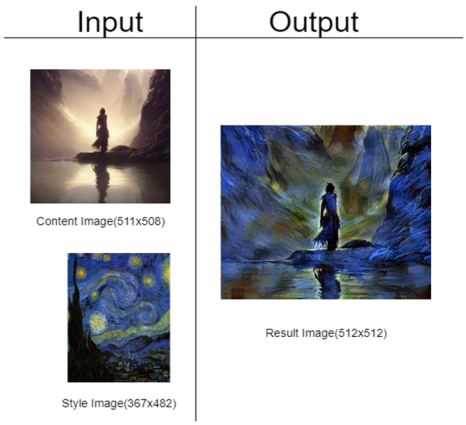


Fig 3. Input and Output example of the Graph Neural Networks

We use a convolutional neural network to do it. The input layer, the convolution layer, and the pooling layer are the three layers that make up the overall convolutional network. Their respective roles also differ from one another. The input layer is used to read the input image; the convolution layer, which is specific to convolution neural networks, is primarily used to extract various features from the input image; the pooling layer's role is primarily for down sampling and dimension reduction;

the removal of redundant data; the compression of features; the simplification of network complexity; the reduction of computation; the reduction of memory consumption, etc. [16].

We implement a neural network that can capture the style feature from sets of training images and apply these styles to perform a style transformation on the object images.

As presenting in Figure 3, this neural network captures the features of the abstract input image. More specifically, it records the RGB value of the image and its structural information in mathematical form. Then it applies those data to the image of the house, transforming it to mimic the abstract image. With similar color and structure, we will feel that the output image contains the style of the abstract image. With different parameters, we can obtain different effects and similarities, just like the six example outputs.

This graph neural network works best for styles with unique structures and distinctive color sets. It can mimic the style of different art schools and famous artists. It can process both image sets or a single image, but more images are preferred for better effects. However, this neural network should be prepared in advance to save time.

3.3 The Reinforcement Learning Teaches AI to Paint

3.3.1 Concept

To train an AI that can paint like a human painter, we implement Reinforcement Learning models. Similar to human painters who paint the image in distinct strokes, this model employs several strokes as agents. That means the original image will be broken down into steps of strokes in order to be re-painted. During each step, a loss function will calculate how close the current painting is to the original image as the reward. Instead of finding the stroke with the maximum current reward, this algorithm calculates a cumulative reward for several steps in order to get the best solution. Figure 4 clarifies the entire process of the design based on the DRL framework.

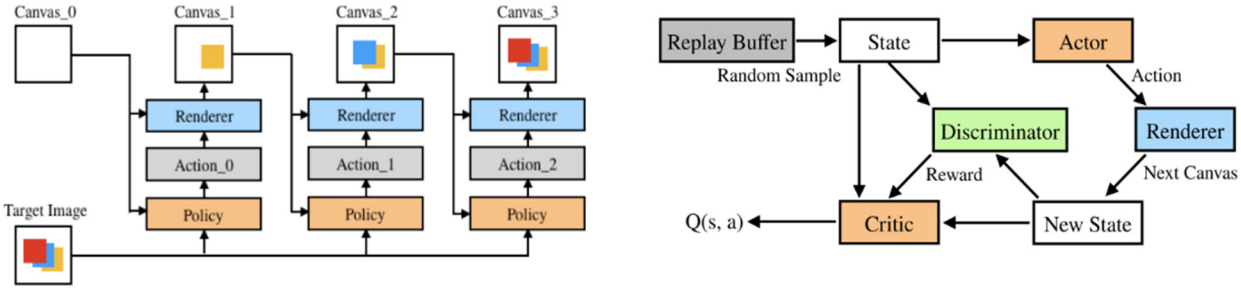


Fig 4. The process of model

We model the issue as a continuous Markov decision-making process with reference to the study by Huang, Z., Zhou, S., & Heng, W. [15]. The agent must determine which set of strokes (a_0, a_1, . , a_{n-1}) may be mapped to C_t to generate C_{t+1} given a target picture I and a blank canvas (C_0). These strokes are mapped out in order to get the final image C_n. A state space S, an action space A, a transition function trans (s_t, a_t), and a reward function r are all present in the problem, as they are in other Markovian decision processes (s_t, a_t).

To determine how much the current output outcome differs from the desired image, we must select an acceptable metric. Here is the prize calculation formula.

$$r(s_t, a_t) = L_t - L_{t+1} \tag{1}$$

where $r(s_t, a_t)$ denotes the incentive for step t, L_t is the measurement error between I and C_t, and L_{t+1} denotes the error between I and C_{t+1}. The discriminator score in this model is set to L.

Throughout the training phase, the agent should work towards maximizing the cumulative reward in order to ensure that the final output results are as similar to the goal picture as feasible. With a

discount factor of $\gamma \in [0, 1]$, the agent's objective at each step is to maximize the discounted total of future rewards is $\sum_{i=t}^{T} \gamma^{(i-t)} r(s_t, a_t)$.

In order to provide more detailed rewards, we employed a neural renderer. We also required to develop a feedback system for the model.

The actor π(s) and the critic Q(s,a) networks were initially established in the DDPG. A policy is modeled by the actor. The critic, educated with information from an empirical playback buffer, employs the Bellman equation to assess the predicted reward produced by the agent's behavior in that state.

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1})) \tag{2}$$

Where $r(s_t, a_t)$ is the reward given by the environment when state st performs an action. The actor $\pi(s_t)$ is trained to maximize the critic's estimate of $Q(s_t, \pi(s_t))$.

3.3.2 Brushes and Transitions

Inspired by the work of Huang's team [15], we began to consider changing the shape of each stroke to change the overall style of the picture. However, there are certain problems with this in practice. We found that the shape of the strokes during the drawing process did have some effect on the drawing style (especially at the beginning), but the overall effect was not significant. Moreover, due to the established reward mechanism of deep reinforcement learning, the results tend to be insignificantly different in style. Therefore, we propose a method to draw pictures with different kinds of brushes.

Due to the low likelihood and difficulty of implementing changes to the deep reinforcement learning reward mechanism, through our research, we decided to achieve an overall change in painting style by migrating each stroke to a specific style. Although it increases the timing complexity somewhat, it does not affect the overall performance.

Several interface functions are provided in OpenCV for non-realistic drawing, including an oil painting style, a watercolor style, image smoothing without affecting edges, image sharpening, an image depiction style, etc. In the testing phase, for each stroke drawn by the function model, we access the above function and do a style transformation on the drawn circular strokes. This can be equated to a human drawing with different brushes, such as watercolors, oil pastels, colored pencils, etc. (Figure 5-6).

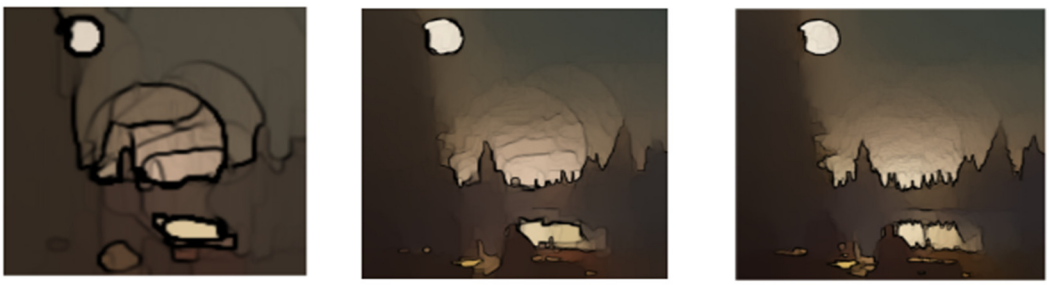


Fig 5. The process of painting in watercolor style

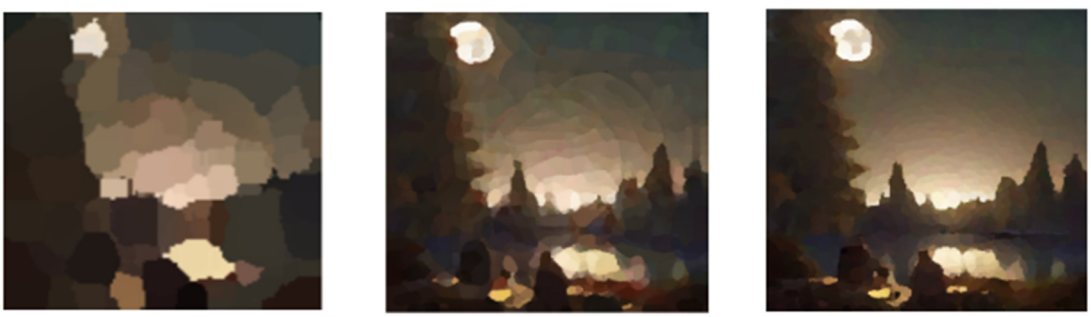


Fig 6. The process of painting in Oil painting style

Through the study of the original model output video and the literature we can find that the rendering of the model is divided into two steps, the first is the rendering of the general framework, and the second is the display of details, but the sudden change from framework to details in the output video is too unnatural, so this thesis thought of adjusting the thick and thin changes in the strokes of the two parts of the transition. However, the thickening of the strokes in the detail part will lead to a low definition of the final image. Therefore, this paper reduced the difference between the thick and thin parts of the strokes and reduced the number of strokes in each step on the basis of increasing the number of steps, and finally we got the result as good as expected, avoiding the problem of the original model.

4. Experiment

4.1 Experiment Platform

Here, Google Colab served as our test platform. It may save a lot of time setting up the environment because it is practical, quick, and effective. Each component requires a separate environment. Diffusers (0.8.0), transformers scipy, and ipywidgets are required for the Stable Diffusion section, while cv2, imageio, os, and numpy are required for the RGB value iteration section. The first two components of AI Artist already contain the environment needed for the third and final portion.

Each component may be trained alone, and the three components' training are mostly independent of one another. We utilize COlab's GPU mode for training since it can significantly reduce training time and increase training effectiveness. We utilized DreamBooth as a training agent for the text to picture component. We took 54 hours, trained 800 landscape-related images, and performed 80,000 steps in total. Naturally, 512 x 512-pixel images are a must for this training, and we also offer fp16 for training. There is not a lot of training needed for the second half. To create a mixed picture after uploading a style image, import the "implementNTS" module, adjust the image size, and execute the style transfer. The important experiment settings are "Number of iterations: 3000, content weight: 1, style weight: 0.01,". However, we may also set additional relevant values. The last step in artificial intelligence painting entails training the neural renderer before importing the learned model. The training sets used in our training are all from the Kaggle website.

4.2 Text to Image

Our project's initial stage involves using stable diffusion V1-4 to produce related images from text descriptions.



Fig 7. Image generates by stable diffusion (pompt: A man fly)

Even though we can't train the complete neural network by ourselves, we can train a tiny portion of it with the help of texture inversion. Each word is initially transformed into a token that represents the matching entry in the model dictionary before being trained. These are subsequently transformed by the dictionary into "embedded" tokens, a continuous vector representation. These embeddings are frequently learned as part of the training procedure. These embeddings are then connected to other sentences and may be added to other sentences at any time. In other words, the text embedding space is reversed.

This connection is made in our research on stable diffusion. Instead of utilizing the trained model to create photographs, we may create pictures in our own manner, using our own training through precise language descriptions (Figure 7).

4.3 RGB Value Iteration of the Image

We iterate RGB pixel values using pictures created by stable diffusion, and we may select different iteration objects depending on our preferences. This allows for the iterative application of the iterated item's image style to the iterated object.

RGB values are used to determine each of the numerous pixels that make up an image. In order to repeat the RGB values of each pixel on the iterated object to the image produced by stable diffusion, we first construct a mathematical model for the iterated object and the iterated object. We may see the presentation of several picture styles in this way.

The size of the iterated object and the iterated object must be adjusted before iteration can begin; this size is often set to 512x512 pixels. Because we will encounter images in various formats while choosing the object to be iterated, we must additionally run a piece of code to convert RGBA to RGB. There are three alpha channels in some images and four in others. If users don't convert beforehand, the iteration will not succeed.

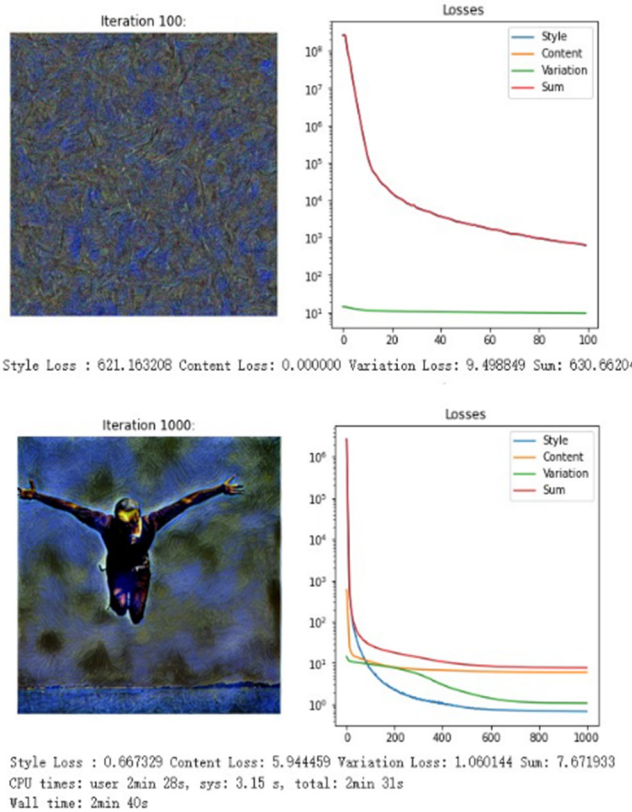


Fig 8. Iterate 1000 times vs. iterate 100 times

When iterating, several parameters (such as "number of iterations," "content weight," and "style weight") may be adjusted, and the iteration's outcome can also vary (Figure 8). The iterated object's pixel value increases with the number of iterations. The object being iterated is nearby.

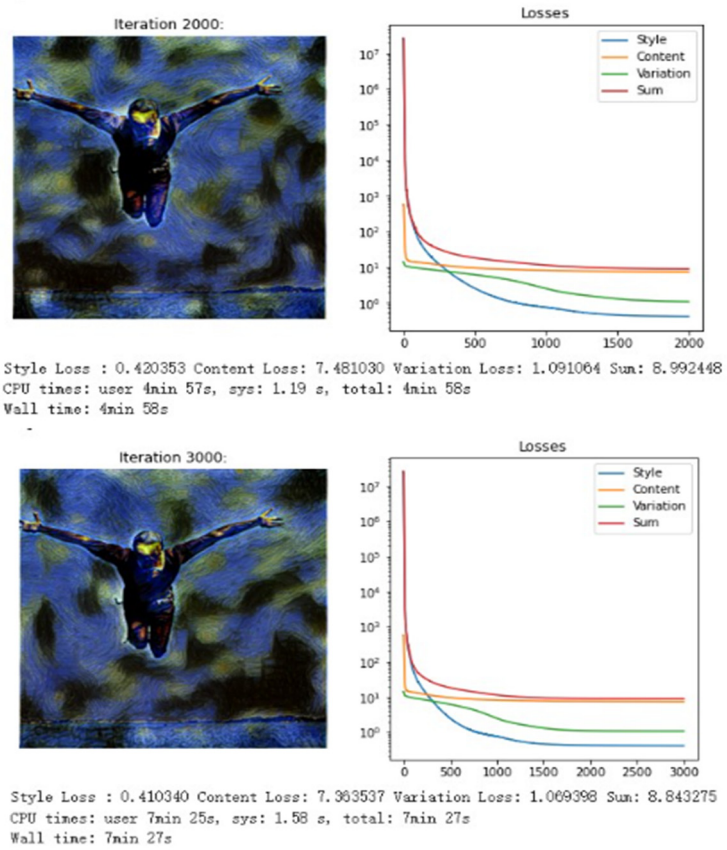


Fig 9. Iterate 2000 times vs iterate 3000 times

To iterate on the findings, we experimented with various settings for eight hours. In order to get the outcomes presented in Figure 8, we started with 1000 iterations, 1 content weight, and 0.1 style weight. The number of iterations was then changed to 2000 and 3000. A closer match between the RGB pixel value of the iteration object and the iteration object is indicated by the loss value in Figure 9: the more iterations, the closer the variation loss of the result is to 1.

By varying the proportion of "content weight" to "style weight," we also attempt to get various outcomes. We discovered that the greatest iterative outcomes occur when the ratio of "content weight" to "style weight" is 10. Other ratio values can typically only determine an iterated object's pixel value, which will destroy the model structure of the iterated object.

4.4 AI painting

After completing the aforementioned steps, we click the final link and allow the AI to draw the images it has generated. A week of training is required, and it will be done using Colab.

Here, we train 3 little agents (critic, actor, and discriminator) using the dataset from Kaggle [18], which contains around 2600 images. The entire agent training procedure is carried out on Colab, and the entire training process takes approximately 73 hours. The other two agents are trained from scratch, with the exception of the discriminator, which makes use of a learned model.

We begin testing the AI painting algorithm after the training. By altering the stroke pattern, we can see the training outcomes. There are two choices that let you paint the brushstroke shape: "Triangle" and "Round." By adding different painting style codes, we enable AI to create images in a variety of styles, just like a street artist can create paintings in a variety of styles using various pigments.

We have added the terms "watercolor," "sketch," and "oil painting" to the list of stroke kinds (Figure 10). Drawing with various strokes requires various amounts of time. In general, sketching takes the shortest amount of time, followed by oil and watercolor painting.

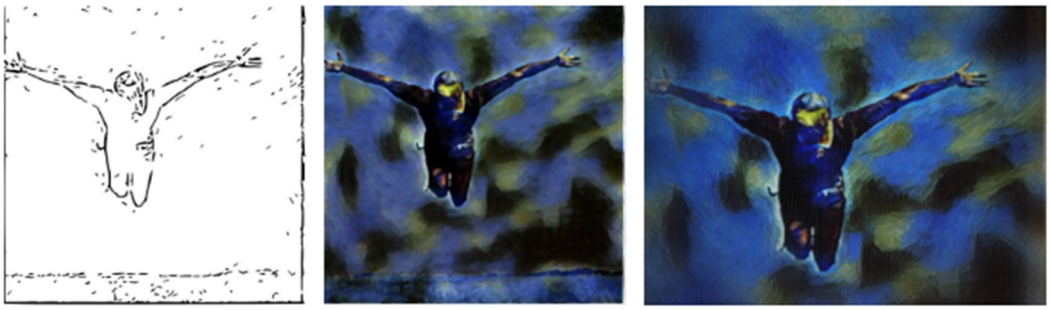


Fig 10. From left to right: sketch, watercolor and oil painting

The figure 11-12 shows the whole process of AI painting.

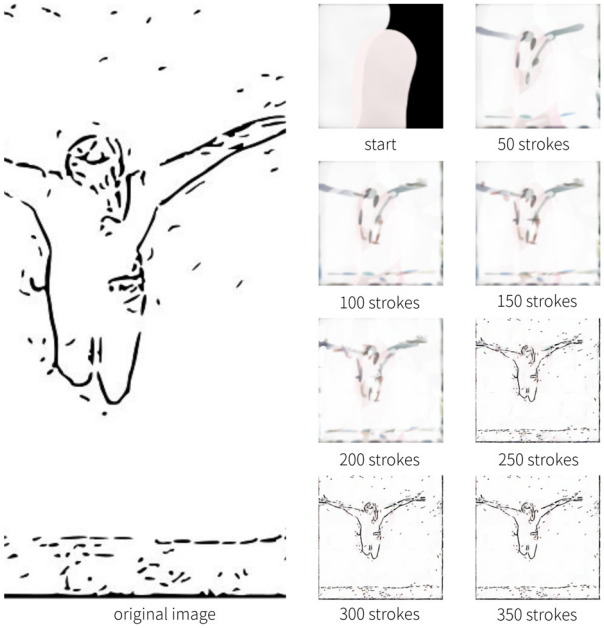


Fig 11. Drawing process of sketch

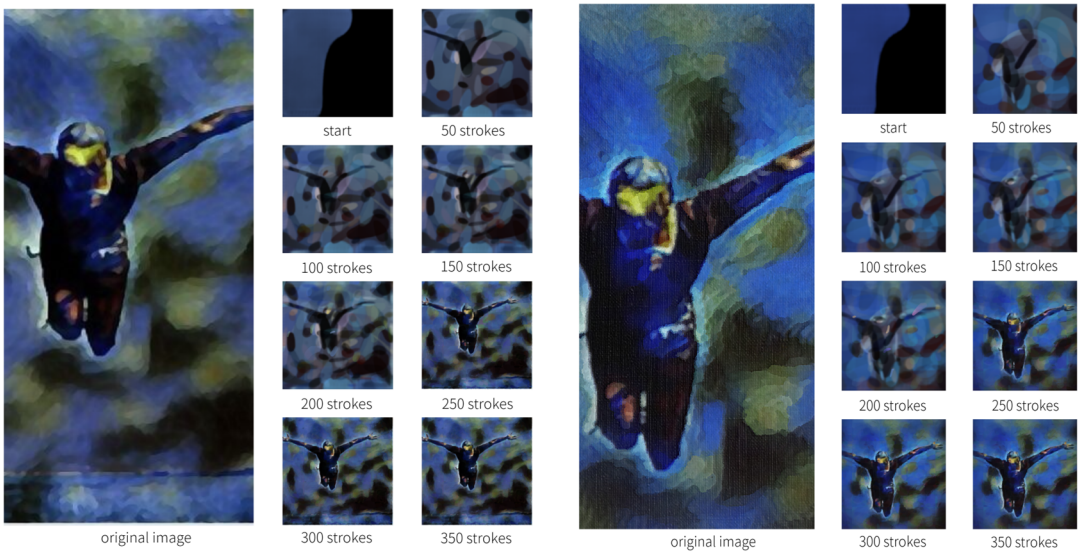


Fig 12. Drawing process of watercolor (Left) and oil painting(right)

As we can see, the color and texture of the strokes is what distinguishes an oil painting from a watercolor or a drawing. They all utilize 350 strokes to paint, which is the same thing, however this

number may be changed. The more strokes used, the closer the painting will be to the original, and the more details it will have.

5. Conclusion

This paper presents a professional and humanized AI artist, improving from the original stable diffusion method. It satisfied the clients' customized requirements for the AI paintings from two aspects, the painting styles, and painting techniques. For painting styles, this AI could mimic famous artists or art schools, especially for their well-known color styles and structure. Besides the style, this AI is also versatile and talented with various painting techniques, fulfilling the clients' need for different textures and methods of painting. Last, like any skilled human painter who provides service in public, it displays the entire painting process as part of the enjoyment. This AI learned to paint like a human, which paints with distinctive strokes, from a vague scratch to a complete masterpiece. In the end, the client can get an ideal painting of their own customized style and a video of the complete painting process instead of a sole image generated from stable diffusion.

Through value iteration, we successfully created graphics with text and altered their style before letting AI draw the results. The framework of reinforcement learning serves as the foundation for the whole training process. The Dream Booth agent performs stable diffusion training. The convolutional neural network is used for style transfer and AI painting training, while the neural renderer is used for the other. Ai painting may also increase image quality by increasing strokes and improving algorithms, and style transfer can do the same by incrementally increasing the number of repeats. We also enhance AI painting using various RGB values for the strokes and create various kinds of images by varying the strokes' application. (Painting in watercolor, oil, or drawing, for example.) The selection of a picture's kind is substantially improved.

References

[1] Khurana, D., Koli, A., Khatter, K., & Singh, S. Natural language processing: State of the art, current trends and challenges. Multimedia Tools and Applications, 2022: 1-32.

[2] Mazzone, M., & Elgammal, A. Art, creativity, and the potential of artificial intelligence. Arts, 2019, 8(1), 26.

[3] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. High-resolution image synthesis with Latent Diffusion Models. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, 1-11.

[4] Cohen, P. (2016). Harold Cohen and AARON. Ai Magazine, 37(4), 63-66.

[5] Little-Tetteh, K., & Shchyhelska, H. Artificial intelligence painting: is it art, really?. 2019, 73-75.

[6] Cetinic, E., & She, J. Understanding and creating art with AI: Review and outlook. ACM Transactions on Multimedia Computing, Communications, and Applications, 2022, 18(2), 1-22.

[7] Liu, Z., Luo, P., Wang, X., & Tang, X. Deep learning face attributes in the wild. 2015 IEEE International Conference on Computer Vision, 2015.425.

[8] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. Generative adversarial networks: An overview. IEEE signal processing magazine, 2018, 35(1), 53-65.

[9] Aggarwal, A., Mittal, M., & Battineni, G. Generative adversarial network: An overview of theory and applications. International Journal of Information Management Data Insights, 2021, 1(1), 100004.

[10] Karras, T., Aittala, M., Aila, T., & Laine, S. Elucidating the Design Space of Diffusion-Based Generative Models. 2022 arXiv preprint arXiv:2206.00364.

[11] Portilla, J., & Simoncelli, E. P. A parametric texture model based on joint statistics of complex wavelet coefficients. International journal of computer vision, 2000, 40(1), 49-70.

[12] Simonyan, K., & Zisserman, A. Very deep convolutional networks for large-scale image recognition. 2014 arXiv preprint arXiv:1409.1556.

[13] Gatys, L., Ecker, A., & Bethge, M. A neural algorithm of artistic style. Journal of Vision, 2016, 16 (12), 326.

[14] Gatys, L., Ecker, A. S., & Bethge, M. Texture synthesis using convolutional neural networks. Advances in neural information processing systems, 2015, 28.

[15] Huang, Z., Zhou, S., & Heng, W. Learning to paint with model-based deep reinforcement learning. 2019 IEEE/CVF International Conference on Computer Vision. 2019, 4378-4389.

[16] Keiron O'Shea, Ryan Nash. An Introduction to Convolutional Neural Networks. arXiv: 1511.08458.

[17] Binxu Wang, Understanding Stable Diffusion from "Scratch", Harvard Medical School, 2020 https:// scholar. harvard.edu/binxuw/classes/machine-learning-scratch/materials/stable-diffusion-scratch.

[18] INNAT, Wiki-Art: Visual Art Encyclopedia, 2022, https://www.kaggle.com/datasets/ipythonx/wikiart-gangogh-creating-art-gan.