Style and Visual Fidelity: An investigation of QuantArt

Yuhan Wang
Smith College
ywang70@smith.edu

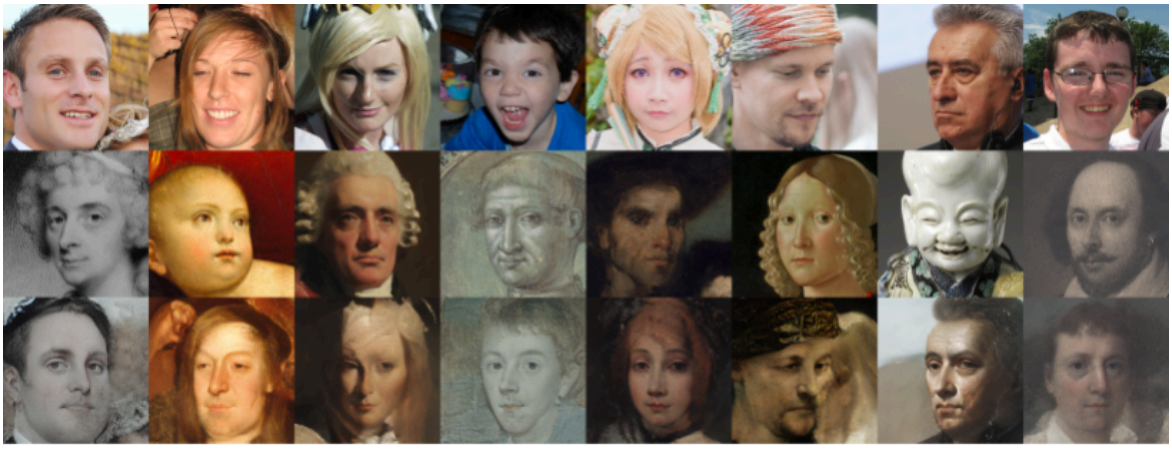

Figure 1. QuantArt style transfer testing results on face to art pretrained models

ABSTRACT

QuantArt is a style transfer framework that combines a content image with an art reference image into a styled production. QuantArt allows arbitrary trade-offs between style fidelity, visual fidelity, and content preservation. This paper investigates the style and visual fidelity of QuantArt, especially on the fact-to-art aspect. I provide insights on the necessary configuration and potential difficulties one might encounter in performing state-of-the-art style transfer models, evaluate the pre-trained model's performance on the original dataset and unseen dataset, and train the model on a new dataset to evaluate the visual fidelity of the style transferred face images.

1. INTRODUCTION

QuantArt is a style transfer framework introduced by Siyu Huang et al. [1] QuantArt allows users to arbitrarily trade off between style similarity and content reservation while maintaining a high "visual fidelity", that the style-transferred image is visually distinguishable from the original content image and visually in accordance with the artistic reference. QuantArt implements vector quantization on the content pictures and the artistic references to create "codebooks." The stylized features would then be pushed closer to the center clusters of the codebooks to preserve content reference and style fidelity. QuantArt also uses Style Guided Attention(SGA) modules to control the style fidelity, facilitating the arbitrary control of style. Their work includes a wide range of style transfer conversions, including photo-to-photo, photo-to-art, art-to-photo, and more sub-categories.

In this paper, I will investigate the style and visual fidelity of QuantArt through experimentations on

its code. First, I will provide a basic overview of the necessary preparation for implementing state-of-the-art style transfer codes in Mac and Windows systems. Secondly, I will focus on the face-to-art aspect of QuantArt and investigate the degree of style and visual fidelity that was highlighted in QuantArt. Finally, from a qualitative perspective, I will train the model on my own dataset and evaluate the final production in the face-to-art category.

2. BACKGROUND

Style Transfer is a computer vision task to combine the style of an artistic image with the content of another source image. Since its first introduction as Neural Style Transfer [2], style transfer has incorporated different state-of-the-art models like GANs [7] and Diffusion Models [4] for image generation. Arbitrary Style Transfer [3, 10] has become a research locus recently because of its ability to combine unseen content-art pairs in a zero-shot manner. Two main trends of style transfer exist, in a heterogeneous manner, to produce more photo-realistic images [5,6] or to generate photos with higher artistic quality [8,9]. The enhancement of photorealism is considered a direct negation of artistic style perseverance [7], and so artistic style transfer and photorealistic style transfer are always performed separately through differently designed models. Existing state-of-the-art models before QuantArt as horizontal comparisons include ArtFlow [8], CAST [9], Stytr2 [10], and AvatarNet [11]. In addition to style transfer, vector-quantized image representations, which were mainly used in the image modeling field [18], are adopted in QuantArt to generate stylized features. QuantArt has achieved better visual fidelity performances from both quantitative comparison, FID score and Gram loss, and qualitative aspects, from human evaluations, and was chosen in this study because of its partially successful configuration on my Macbook Air 13.

QuanArt's framework uses 4 auto-encoders to extract both continuous and quantized features from content and artwork images respectively, obtains 2 codebooks to store the distributions of content and artwork images, and 2 SGA modules to transfer the feature representations. [1] The training of QuantArt consists of two stages. In the first training stage, we learn 4 auto-encoders and 2 codebooks by reconstructing the photo and artwork images. In the second training stage, we train the SGA modules based on the extracted feature representations. In the inference phase, we can trade off the style and visual fidelity of generations by adjusting the discretization level $\alpha, \beta \in [0, 1]$ between the SGA outputs and before the final image generation.

While QuantArt and its horizontal comparison models were all introduced in very recent years, they have implemented drastically different dependencies for Python, torch, torchvision, pytorch-lightning, cuda toolkits, etc. The unprecedentedly rapid development of deep learning, meanwhile, has pushed successive new versions of computer vision Python libraries [12]. While newer releases usually provide much easier and more integrated implementations, functions from packages with an older version are often overwritten or removed. Researchers who want to implement older models thus suffer from dependency checks and search for old-versioned packages. In addition to issues in compatibility, the computer vision field, in general, has been directed towards more powerful

implementations that require larger training datasets, more GPUs, and longer time, which might inhibit fast, simple experiments with state-of-the-art works [13, 14]. QuantArt and other recently introduced style transfer models, for example, were commonly trained on large datasets like COCO [15], WikiArt [16], and FFHQ [6], and implemented transfer learning from large pre-trained models like VGG [17], requiring large amounts of time and effort for researchers to simply replicate their work.

3. PROJECT DESCRIPTION

3.1 Dependency & Configuration

To implement QuantArt's code in a local environment, the first necessary step is to match the environment based on the dependencies. All the used packages are not the latest versions, so it is necessary to downgrade them to their proper versions. Since QuantArt recommends building a Conda environment for the model, it is easier to use Anaconda to perform the downgrading of all the packages. The downgrading process would also encounter immediate conflicts with other existing packages in the computer, so it is necessary to downgrade the unused packages as well to ensure that all requested packages are in the right versions.

Based on two crucial packages in QuantArt's dependencies– Cuda toolkits 10.2 and pytorch-lightning 1.0.8– it can be inferred that QuantArt was trained in a Linux environment with 4 GPUs. Cuda toolkits 10.2 was included to facilitate the usage of multiple GPU units on the local machine to accelerate the training process. For Mac users, Cuda toolkit 10.2 can be downloaded through NVIDIA but is not compatible with the Mac OS system or with Apple's M2 chip. Thus, for Mac systems, the code can only be run on the CPU without using Cuda. For users who have GPU units on Windows, Cuda toolkit 10.2 is compatible with the system but not the GPU. The accelerator that pytorch-lightning 1.0.8 uses in QuantArt, "ddp", does not support Windows systems. Changing the accelerator to "dp" can solve this problem but will raise NotImplemented errors. Installing a WSL (Windows Subsystem for Linux) on Windows can solve the accelerator problem, but as Cuda toolkit 10.2 does not support the WSL system, this option will eventually work the same way as a Mac OS system. Therefore, for Windows users with GPU, the code can be run directly on Cuda with GPU for testing, on Cuda without GPU for training, and on CPU if using the WSL system.

3.2 Testing on existing dataset and my dataset

For the training and testing, QuantArt uses public datasets like COCO [15], WikiArt [16], FFHQ [6], and MetFaces [19]. These datasets need to be downloaded and stored in the datasets directory with proper labeling according to QuantArt's documentation. Most training and testing processes will only require 256 images from the dataset, so it is not necessary to store the entire dataset with 30k photos on a local machine. In addition, I have created a dataset of 200 photos of my face as the new dataset. All photos are renamed in accordance with the art reference dataset and reshaped into 1024 x 1024 pixels. Pretrained weights from each stage of QuantArt are publicly available, so users can choose to

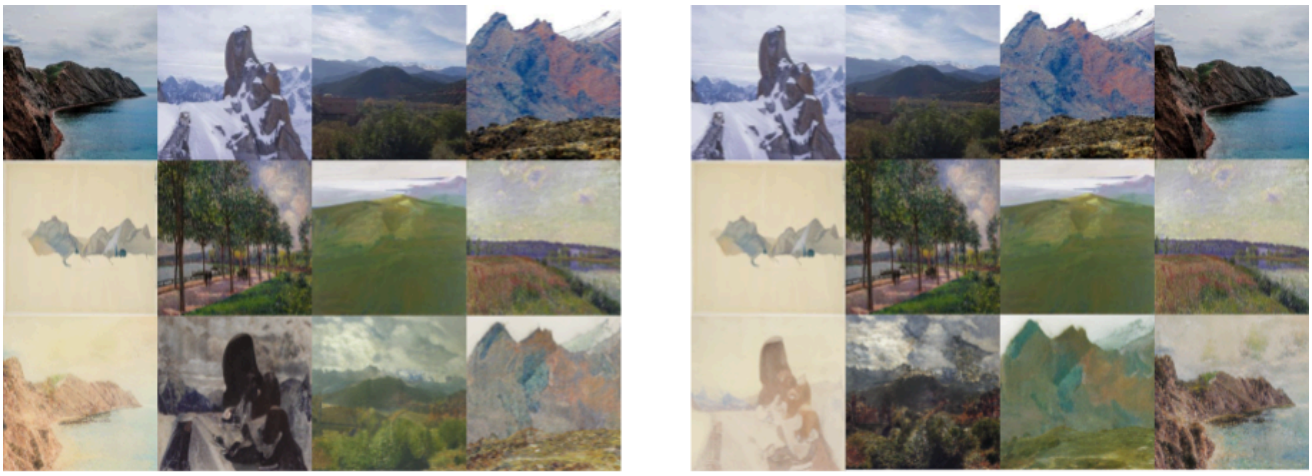

Figure 2. Quick testing of Style Transfer on landscape to art. Landscape-artwork pairs are shifted in the 2 testings.

experiment with stage 1 for the training of autoencoders and codebooks, stage 2 for the training of SGA modules for style transfer, and inference stage to test the style transfer on seen and unseen data.

I start my experiments with base testing, which is a quick style transfer on four example landscape artwork pairs. (see Figure 2) Since the base cases are workable, and the style transfer results are visually viable, I perform testing on the face-to-art dataset. The reason for choosing face-to-art for this project is that humans have a stronger ability and higher standard for face images, and it is often more difficult for machines to generate images for faces. [20] The testing on the face-to-art dataset requires a transfer learning process for the final inference stage. Because my computer does not have enough storage, I exemplified 1000 images separately from the face dataset and the metface dataset. The transfer learning runs 125 epochs, which takes 5 hours to complete only using the CPU. The inference stage eventually produces 128 photos with 8 face-art pairs and their style transferred images. (see Figure 1) In addition to the style transferred images, it also outputs the configurations used for further testing. With those configuration files, I run a quick test on my photos to test how it expands on unseen data. (see Figure 3)

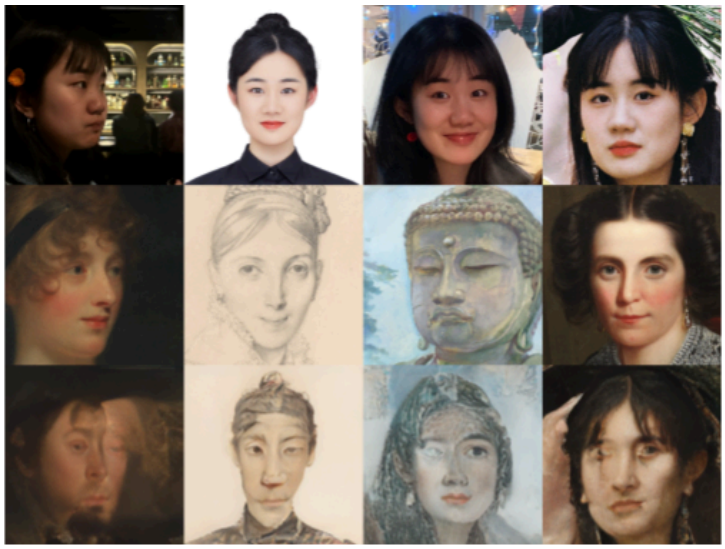

Figure 3. Testing on face to art dataset with unseen photos

3.3 Training on my dataset

After testing the quick transfer learning from the last stage, I perform the stage 2 training on my own dataset. Stage 2 uses the already trained autoencoders, codebooks, and content-art pairs to train SGA modules, which will facilitate the style transfer process specific to the face-art pairs. Because only CPU computing is available due to machine limitations, I only experiment with the stage 2 training under the scope of this project. I test the training on a small patch (16 images) and the whole dataset (200 images) respectively. QuantArt offers instructions for both training and testing, so I create a new config file for my dataset and use the given command to perform both training. However, no specific codes or instructions are found to change the training epoch, tune the α, and β parameters, or perform quantitative testing on the generated results. Therefore, experiments are all performed with default settings, and evaluations are mainly visual analyses.

The training of stage 2 with 128 epochs on the 200-image whole dataset in Windows using Cuda takes 2 days to finish. Each epoch saves the last checkpoint record, and after around 30 epochs the top three parameters of the epochs with the smallest loss are saved as well. The loss starts from 12.9 to 0.38 around 30 epochs, but quickly rises to 1.2 around 60 epochs. It is very possible that overfitting occurs because of the small dataset. Images of training and validation results are saved for the last epoch, and deleted during the next epoch to save space.

4. EVALUATIONS

4.1 Pretrained model evaluation

As Figure 1 and Figure 4 show, the face-to-art pretrained models' performance is not as perfect. The style elements injected into the final production and the reference to the content images are visually consistent. However, the visual fidelity of the final productions is unstable. We notice some blurry facial features in the style transferred images. Particularly, there are two types of conditions that might lead to the failure of style transfer. The first case is when the content image and the art image have different face orientations or very different angles, for example, one front view and one side view. The different face orientations might result in an unnatural combination of facial features on a face contour.

Figure 4. Style transfer results from pretrained face to art model on FFHQ [6] dataset.

Similarly, some face decorations like glasses are particularly unclear or confusing in the transferred productions. The second case is more general that style transfer typically performs better on certain ethnicities. This could be due to the bias of the metface dataset. While the dataset does contain a good number of artworks from different cultures, realistic, natural face portraits are mainly from traditional Western paintings. Many non-Western faces are from idealized architectures or symbolized paintings, which might cause an uneven representation for all ethnicities.

4.2 Training evaluation

Figure 5. Face to art transfer learning on my dataset trained on 16 images

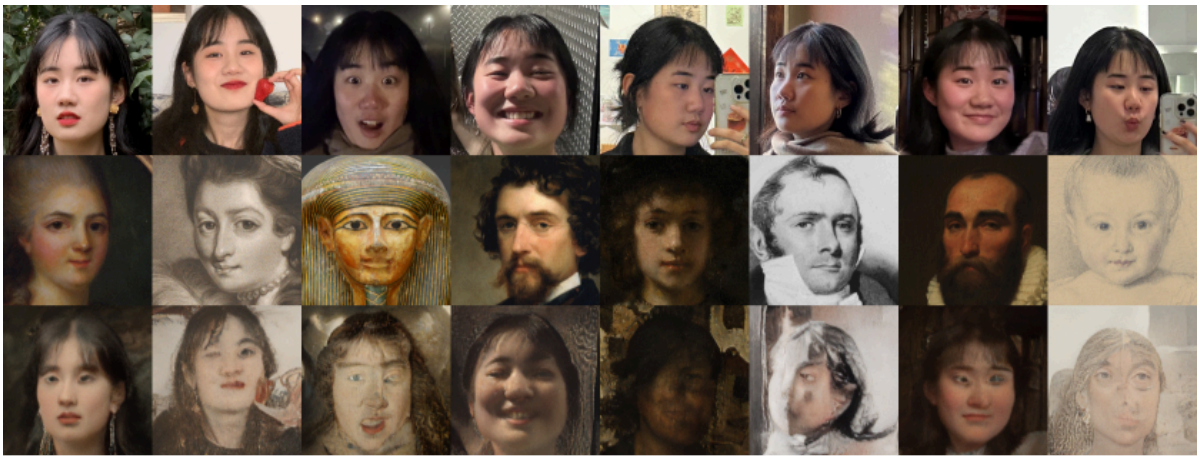

Figure 6. Face to art transfer learning on my dataset trained on 200 images

From Figures 5 and 6, we can see that training stage 2 on the unseen dataset produces better transfer learning results, especially for the face contour synthesis. The training takes a long time because of the GPU's incompatibility. Training the sampled dataset with 16 images on Mac OS with CPU takes 20 minutes for each epoch; training the whole dataset with 200 images on Windows with Cuda and CPU similarly takes 20 minutes for each epoch. This proves that using Cuda as the computation unit,

though without GPU, has largely accelerated the training process. We can infer that with 4 GPU units as described in the paper, the training process will be even faster since every training only takes 256 images.

Comparing Figure 1, Figure 5, and Figure 6, we can see that the training on SGA modules has facilitated the preservation of both information from the content images and styles from the reference images. Comparing Figure 5 to Figure 1, the face contours in Figure 5 are more clearly defined, visually corresponding to the content image. The tone and filter-like style of the generated images are highly similar to the art images. This discovery proves the functionality of SGA modules as described in the paper, to enhance additional self attention to distinctly preserve the content feature and style reference. As the α, β parameters that control the visual and style fidelity are put in between SGA units in the QuantArt framework, future experiments can focus on the potential changing of α, β to evaluate the quantification of the content feature and style reference accordingly.

Comparing Figure 6 to Figure 5, we observe that a larger dataset will improve the generation of facial features to be more detailed and visually explainable. While Figure 5 shows that SGA modules preserve the general shape of the faces, the facial features are yet blurry and sometimes contain references more relevant to the art images. In Figure 6, however, we see that the generated images are more clearly defined with more apparent style references. Since GPU units are not used, and the codebook also lacks my dataset in the distribution, it is hopeful that the implementation of the whole training process in the recommended environment will yield successful style transfer results with high style and visual fidelity.

Based on the contents above, the ability to perform style transfer on new datasets and the effectiveness of SGA modules have been proven through the experiments. In the future, potential directions include: configuring a Linux environment with 4 GPUs to perform the whole training, adding tuning of epochs to test on small datasets, modifying the α, β parameters to compare the style and visual fidelity of the generated images, and leverage evaluation metrics to get quantitative results aside from visual analysis. State-of-the-art models after QuantArt have achieved better visual results in a zero-shot manner, while later Cuda toolkits also have the potential to run on WSL systems. Therefore, for QuantArt, the tuning of α, β to control the style elements might be the most interesting research focus for future implementations.

5. CONCLUSION

In this paper, I experiment with QuantArt in the fact-to-art category. I evaluate the required configuration and potential difficulties in implementing state-of-the-art style transfer models in Mac and Windows environments. I also investigate the potential features in the content and the bias in the art dataset that might cause failures of style transfer. Through experiments on pre-trained models and training on stage 2, I show the effectiveness of SGA modules and the effect of a larger dataset to deblur the generated images. While some style-transferred images display inconsistent facial features,

most images convey consistent, visually explainable faces, with the contours being clearly defined and artistic styles being preserved.

6. REFERENCES

[1] S. Huang, J. An, D. Wei, J. Luo and H. Pfister, "QuantArt: Quantizing Image Style Transfer Towards High Visual Fidelity," 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 2023, pp. 5947-5956, doi: 10.1109/CVPR52729.2023.00576.

[2] Gatys, Leon A.; Ecker, Alexander S.; Bethge, Matthias (26 August 2015). "A Neural Algorithm of Artistic Style". arXiv:1508.06576 [cs.CV].

[3] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In CVPR, pages 1501–1510, 2017.

[4] Lanqing Guo, Chong Wang, Wenhan Yang, Siyu Huang, Yufei Wang, Hanspeter Pfister, and Bihan Wen. Shadowd- iffusion: When degradation prior meets diffusion model for shadow removal. arXiv preprint arXiv:2212.04711, 2022

[5] Jie An, Haoyi Xiong, Jun Huan, and Jiebo Luo. Ultrafast photorealistic style transfer via neural architecture search. In AAAI, volume 34, pages 10443–10450, 2020.

[6] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative Adversarial Networks,” arXiv.org, https://arxiv.org/abs/1812.04948 (accessed Apr. 28, 2024).

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle- consistent adversarial networks. In ICCV, pages 2223–2232, 2017

[8] JieAn,SiyuHuang,YibingSong,DejingDou,WeiLiu,and Jiebo Luo. Artflow: Unbiased image style transfer via re- versible neural flows. In CVPR, pages 862–871, 2021.

[9] Yuxin Zhang, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, Tong-Yee Lee, and Changsheng Xu. Domain enhanced arbitrary image style transfer via contrastive learning. arXiv preprint arXiv:2205.09542, 2022.

[10] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr2: Image style transfer with transformers. In CVPR, pages 11326– 11336, 2022.

[11] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar- net: Multi-scale zero-shot style transfer by feature decora- tion. In CVPR, pages 8242–8250, 2018.

[12] “PyTorch,” www.pytorch.org. https://pytorch.org/blog/pytorch-2.0-release/

[13] R. Inc, “AI Vision Software: Cost To Develop A Computer Vision App,” risingmax.com. https://risingmax.com/blog/ai-vision-software-development-cost

[14] G. Boesch, “What Does Computer Vision Cost? An Ultimate Guide for Businesses,” viso.ai, Mar. 26, 2022. https://viso.ai/computer-vision/cost-of-computer-vision/

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dolla´r, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV,

pages 740–755, 2014

[16] K Nichol. Painter by numbers, wikiart. https://www. kaggle.com/c/painter-by-numbers, 2016.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 4

[18] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In CVPR, pages 12873–12883, 2021.

[19] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative ad- versarial networks with limited data. NeurIPS, 33:12104– 12114, 2020.

[20] S. Dooley et al., “Comparing Human and Machine Bias in Face Recognition,” arXiv:2110.08396 [cs], Oct. 2021, Available: https://arxiv.org/abs/2110.08396