Realistic Style Transfer Using Related Images

Medya Tekes Mizrakli (342041), Ekaterina Trimbach (337625), Colin Pelletier (336438)
CS-503 Final Project Report

***Abstract*—As humans, we are able to imagine what a painting looks like in the real world. inferring from our previous encounters with the object. In a way, our brains automatically remove the artistic features from the painting and unveil the real object beneath. In this work, we took inspiration from this process and wanted to mimic it through neural networks by leveraging style transfer networks and choosing landmark images as our playground. We provide both quantitative and qualitative results of our approach and draw conclusions from them.**

I. Introduction

As humans, when we see a drawing or a painting of an object we know, we are able to interpret and imagine how it looks like in real world. We are able to do so because we have prior knowledge about it, due to the fact that we have previously encountered with it either in person or through a real image. In a way, our brains automatically remove the artistic features from the painting and unveil the real object beneath. In our work, we took inspiration from this process and wanted to mimic it through neural networks.

The problem of transferring artistic style into real images has been widely studied and provided successful results, such as [1], where a content image and a style image are provided into a style transfer network to create an artistically stylized image. Combining this powerful tool with our initial inspiration, we have reversed the purpose of AdaIN[1] style transfer network and defined our problem statement as "Can we use AdaIN [1] style transfer network to create realistic images from paintings using other related real-life images as reference".

Having a solution for this problem have many practical use cases such as animation, art interpretation, urban planning, advertising, interior design, and many more. By applying realistic textures, lighting, and color grading to their designs, businesses can create more immersive and engaging visuals that draw in viewers and customers. Although the applications of our solution can span through many fields including the ones we mentioned, in the scope of this project we turn our focus to art interpretation. For example, Figure 1a illustrates a painting from 1924 by Charles Mills Sheldon who painted the Hanging Gardens of Babylon which existed around 600 BC. Since this landmark or any images of it does not exist today, we do not know what these gardens looked like at the time. We try to provide a solution where we can use Sheldon's painting and another historical garden, such as the Alcazar Gardens in Seville, shown in Figure 1b as a reference point, to obtain a realistic image to understand what the artist imagined the Babylonian Gardens looked like.

To provide this solution, we created our own stylized landmark image dataset and retrain a style transfer model to teach realistic style transfer and we provide details in section III. The reason we selected landmarks as our playground is the fact that they are one of a kind and everyone has a shared knowledge and understanding of landmarks, hence we can obtain an unbiased and fair qualitative assessment of the outputs from our proposed solution using a survey. In addition to a qualitative assessment through user survey, we performed a quantitative assessment by comparing our model's performance with the baseline model performance on a test set and reported in section IV. Drawing from our experiment results, we end with a conclusion and discussion on the limitations of our solution in section V. Our implementation is available in a GitHub repository [2].

II. Related Work

We used the network architecture proposed by [1] for arbitrary style transfer, but we twisted the problem to focus specifically on creating a realistic image from a stylized image. A similar problem, de-styling a stylized image, was covered by [3], where they proposed to encrypt the content features into the stylized image using stenography, so that the content image can be decrypted from it. This brings the obligation of having the stylized image created along with encryption, which is not possible, especially for hand-drawn paintings. We try to impose reality without having an encrypted initial image so that our model is applicable to all stylized images whether it is created using stenography or not.

Another solution was proposed in [4], where segmentation maps were created for paintings and patches were replaced with realistic patches from their memory banks. However, in our approach, we try to provide a solution that does not rely on a specific memory bank but instead interprets a realistic style using the reference image provided by the user.

III. Method

Our aim is to build a style transfer model that can create realistic images from paintings using a reference image. For this, our approach is divided into two main steps and is visualized in Figure 2.

1) Creating stylized images for the training, validation and test sets using the AdaIN style transfer network

(a) Painting of Hanging Gardens of Babylon by Charles Mills Sheldon (1924)

(b) Seville Alcazar Gardens.

Figure 1: An example of relevant and irrelevant images in the context of landmark images.

[1] with landmark images from "Google Landmarks Dataset v2"(GLD) [5] as the content images and style images from "WikiArt" database [6]. This part is depicted on the left side of Figure 2.

2) Retraining this network to teach the "realistic style" transfer, using the stylized images created in step-1 as the content image and other images of the same landmark category from GLD [5] as the style image. This part is depicted on the right side of Figure 2. Our aim is not to completely recover the original image but rather to make the model learn a realistic style to create a realistic image. Therefore, in the retraining phase, we use different images of the same landmark as the style image instead of the original image. Also, since the original image will not be available to the end user, our approach will enable users to create a realistic image of the stylized image by only using itself and a reference image they think is relevant, as in the Babylonian Gardens and Alcazar Gardens pairing.

A. *Obtaining and Arranging Landmark Images*

GLD [5] includes approximately 5 million images compiled from "Wiki Loves Monuments" (the Wikipedia photo contest around cultural heritage) and "Wikimedia Commons". The information on the landmark categories and image ids are provided in several csv files under 3 main categories: train, index and test. From these, we selected to work with the "train_clean.csv" which is provided by the winning team of the Google Landmark Retrieval 2019 Kaggle Challenge. In this file, for each landmark category all the **relevant** image ids are listed using their retrieval algorithm, which minimizes the number of irrelevant images. An example of these relevant and irrelevant images is provided in Figure 3

From this set, we performed regular data cleaning and to select images of popular landmark categories we removed all the image ids which belong to a category that has less than 150 images. Then, to create a balanced training set we randomly selected 200 landmark categories and 25 images per category, which resulted in 5000 landmark images. For the validation and test sets, we used the same 200 categories as in the training set, but also to be able to make sure that the model performs well on unseen landmarks as well, we randomly selected additional 10 categories and sampled 5 images per category, resulting in 1050 landmark images each for validation and test.

B. *Obtaining and Arranging Style Images*

The WikiArt dataset contains over 100,000 style images. This allows us to incorporate information from many different styles into our model. We created a script to generate landmark&style image pairs, following a specific procedure.

First, we defined three **fixed** style groups:

1) A group of 10 fixed styles paired with all landmark images in the training, validation and test set.
2) A group of 5 fixed styles paired with all landmark images in the validation set.
3) A group of 5 fixed styles paired with all landmark images in the test set.

In addition to these **fixed** styles, we created three **randomized** style groups that do not overlap with the **fixed** styles:

1) For each landmark image in the training set 10 random styles were paired.
2) For each landmark image in the validation set 5 random styles were paired.
3) For each landmark image in the test set 5 random styles were paired.

In the end, each landmark image is paired with 20 style images. We have introduced the random styles to have stochasticity and enable generalization to various styles, and kept the groups of fixed styles to have our model converge.

C. *Creating the Stylized Images*

After obtaining and arranging images we created a stylized image for every pair of *content(landmark)* & *style* using pytorch-AdaIN implementation. This implementation

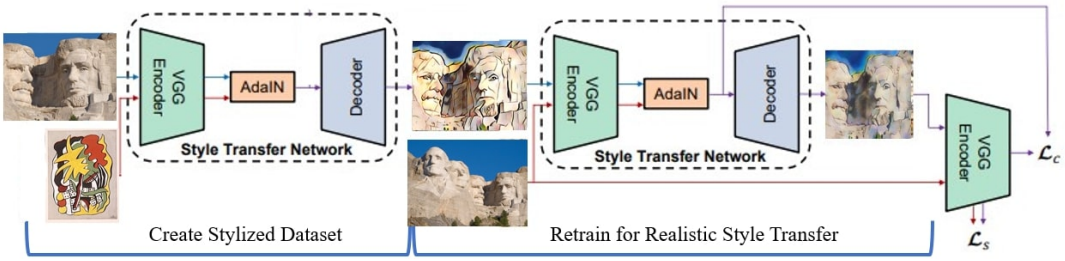
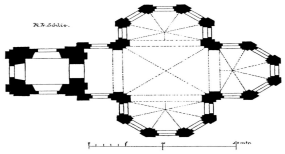


Figure 2: Illustration of the process. First step: creating the stylized image dataset. Second step: retraining the network to teach realistic style transfer.

(a) A relevant image of Krasnogvardeysky Bridge.



(b) An irrelevant image labeled as Krasnogvardeysky Bridge.

Figure 3: An example of relevant and irrelevant images in the context of landmark images.

already contains all model weights and does not require retraining for creating the stylized image. This process corresponds to the "Create Stylized Dataset" part in Figure 2. The model includes the fixed VGG19-encoder layers, an adaptive instance normalization layer and a trained decoder. It takes the RGB landmark image and RGB stylized image as input and returns the stylized landmark image as output to be used in the "Retrain for realistic style transfer" part. In total we were supposed to have 100.000 stylized landmark images in our training set (5000 $landmark\ images \times 20\ styles$), 2.100 stylized landmark images in our validation set (1050 $landmark\ images \times 20\ styles$) and 2.100 stylized landmark images in our test set (1050 $landmark\ images \times 20\ styles$). Due to reasons such as connection interruptions, processing errors, and non-confirming formats, in the end our training set included 64.245 stylized landmark images, the validation set included 20.391 stylized landmark images, and the test set included 20.626 stylized landmark images.

D. Retraining for Realistic Style Transfer

After creating the stylized landmark images, each of them was paired with several other real landmark images. For example, in Figure 4 you can see that the stylized image of Sydney Opera House is paired with 5 other real images of Sydney Opera House. In the end the number of pairs we have created is as follows:

1) Each stylized landmark image in training set is paired with 5 real landmark images, hence a total of 321.225 training pairs.
2) Each stylized landmark image in validation set is paired with 2 real landmark images, hence a total of 40.782 validation pairs.
3) Each stylized landmark image in test set is paired with 2 real landmark images, hence a total of 41.252 test pairs.

IV. Experiments

In this section, we present the different experiments we performed to generate realistic images and the results we obtained.

A. Retraining

For retraining the model we use pytorch implementation of AdaIN [1] and adapt it for our needs. This network uses a basic encoder-decoder architecture and for every image it calculates both the content and style losses as MSE between output features and content and style image features respectively. The final loss is the weighted sum of content and style losses, where the content loss is weighted by 1 and style loss by 10. The training and validation evaluation is described on Figure 5.

B. Baseline

To have a baseline result to compare with our trained model, we used the original AdaIN [1] model without retraining it. First we have visually observed outputs of the retrained model and baseline model by randomly selecting images from the training set pairs. Some examples of the baseline model outputs are presented in Figure 6. Then using our test set pairs, we have computed both the content loss and the style loss for each pair and visually observed some of the outputs.

C. Quantitative Comparison of Baseline and Retrained model

Table I presents the average content loss and style loss over the test set, for each model.

-	**Content Loss**	**Style Loss**
Baseline	0.77 ± 0.30	0.43 + 0.46
Trained	0.82 ± 0.31	0.24 ± 0.25

Table I: Comparison between trained and baseline models.

We note that the baseline content loss is slightly smaller than for the trained model. However, the trained model generates a much smaller style loss.

(a) Stylized Image (b) Paired Image 1 (c) Paired Image 2 (d) Paired Image 3 (e) Paired Image 4 (f) Paired Image 5

Figure 4: Stylized image of Sydney Opera House and paired real images for retraining.

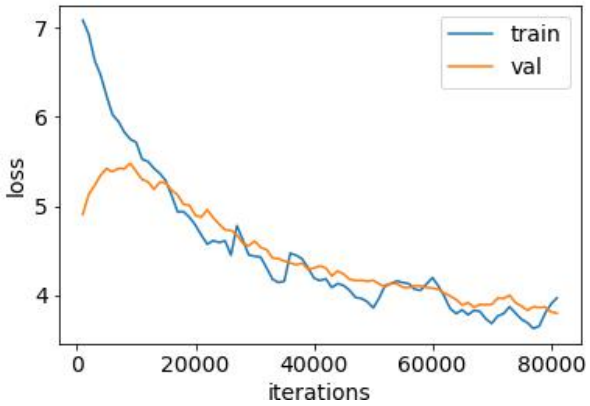
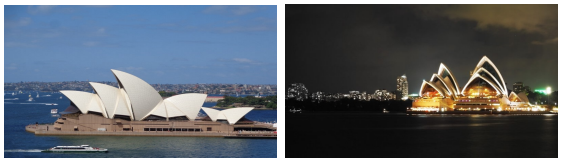
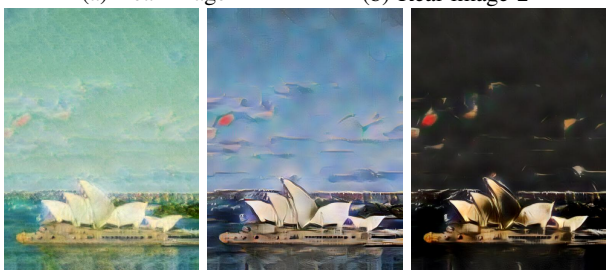


Figure 5: Train and validation loss.



(a) Real image-1 (b) Real image-2



(c) Stylized image (d) Real image-1 output (e) Real image-2 output

Figure 6: Two examples of baseline outputs for stylized image of Sydney Opera House (c) as the content input and paired real images (a),(b) as the style input.

Also, we used t-test to compare distributions for baseline and trained models. The t-test for content loss has p-value 1.038e-135 and for the style loss with alternative hypothesis that trained loss less than baseline p-value equal 1. Hence, we can conclude that we did not significantly changed content loss, but decreased style loss during the training.

D. Stable Diffusion with ControlNet

We also use Stable Diffusion [7] and ControlNet [8] to produce realistic images. ControlNet allows to control diffusion model by adding extra conditions. We considered the Canny ControlNet, which uses the canny edges of the stylized image alongside the prompt to produce the realistic image.

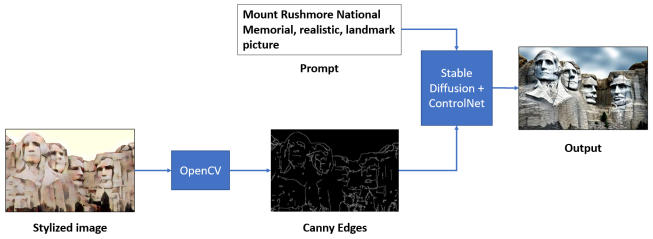


Figure 7: ControlNet pipeline to generate a realistic image

Figure 7 shows an example of the pipeline. To generate a realistic rendering of a stylized image, we first extract the Canny edges of this image using OpenCV [9]. Then, we define the image prompt as "{landmark category}, realistic, landmark picture", where category corresponds to a short description of the image. Finally, we provide both the prompt and the Canny edges to ControlNet.

E. Qualitative Comparison

We conducted a user survey to evaluate the qualitative performances of the different models. The survey contains 10 multiple choice questions, in which the users must select which image(s) look realistic. The images proposed correspond to the output of the baseline AdaIN model, the trained AdaIN model (our implementation), and Stable Diffusion. If none of the image seem realistic, the users can select "None".

Table II contains the percentage of answers which defined the model output as realistic. The percentages do not add upto 100 because users were allowed to select multiple choices if they found them to be realistic.

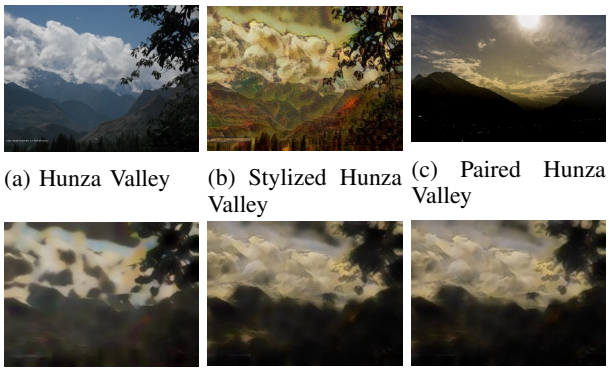
We see that our model slightly improves the baseline, which is consistent with the quantitative comparison results. However, most images from the AdaIN network are defined as not realistic. We note that 52% of the images produced by Stable Diffusion do not look realistic either.

Model	Percentage
Baseline AdaIN	9
Baseline AdaIN	11
Stable Diffusion	52
Not realistic	34

Table II: User survey results. The percentage corresponds to the percentage of questions for which the output images have been defined as realistic.

F. Model diagnostic

To diagnose the poor visuals of our model, we have created a smaller training set consisting of only 230 images and tried to overfit our model to this set to see if we can observe any improvements on the produced results. Figure 8 displays the evolution of the outputs for baseline 8d, iteration number 12000 8e, and iteration number 32000 8f. Due to time restrictions, we could not have the model trained until it overfits, however, the results seen from baseline until iteration number 32000 still give us some insights. Here we see that while style artifacts on the sky improve by further training, we lose details on the leaves and the mountains do not seem to be improving. This shows that although our style loss is decreasing the improvement in terms of getting closer to a realistic image is very slow and not uniform through different segments. Using a different loss function instead of L2 loss or changing weights might remedy the situation however due to time restrictions we were not able to further test those.



(a) Hunza Valley (b) Stylized Hunza Valley (c) Paired Hunza Valley

(d) Baseline result for Hunza Valley (e) $12k^{th}$ iteration result for Hunza Valley (f) $32k^{th}$ iteration result for Hunza Valley

Figure 8: Evolution of output image of stylized Hunza Valley from baseline to iteration 12000 and 32000 using small training set.

V. Conclusion and Limitations

In this project, in an attempt to obtain a model for realistic style transfer, first we generated a stylized image dataset with the AdaIN style transfer network using landmark images from GLD [5] and WikiArt database [6]. We then defined a specific train schema to focus on realistic style transfer. This implementation improves on the baseline version of the AdaIN network in terms of style loss. Although our model has significantly lower style loss on the test set, the visual outcomes of our implementation still remain distant from being realistic. To diagnose the reason we have tried to overfit our model by retraining on a smaller subset, however, due to time restrictions we were not able to train until the model has overfitted. Nevertheless, the evolution of the results from baseline model output to the iteration number 32000 output have shown that the improvement by continuing to train is questionable. From that, we deduce that the L2 loss used for style transfer might not be adaptable for "realistic style transfer" task. The user study showed that even Stable Diffusion fails to generate realistic images in a consistent manner. This gives clear indicators that the definition of "realistic style" is not straightforward and cannot be defined as a unique texture or style, but rather a combination of multiple complex styles that interacts with each other. In order to capture a more "realistic texture", we recommend further research to investigate a multi-ControlNet architecture, using Shuffle and Canny Controlnet. Using Shuffle should allow to capture the texture of the image, while Canny would help keep a coherent and close-to-reality structure in the output image.

VI. Individual contributions

In the fisrst part of the project, landmark images were selected and downloaded by Medya Tekes Mizrakli. Style images were selected and downloaded by Colin Pelletier. Stylized images were created by Ekaterina Trimbach. In the second part, retraining was done by Ekaterina Trimbach, baseline tests and creation of survey images were done by Medya Tekes Mizrakli, Stable Diffusion+ControlNet images and user survey was created by Colin Pelletier. It is fair to say the work load was divided equally.

References

[1] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization." *IEEE International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://github.com/xunhuang1995/AdaIN-style

[2] C. P. Medya Tekes Mizrakli, Ekaterina Trimbach, "Realistic style transfer using related images," 2023. [Online]. Available: https://github.com/Katerina5649/pytorch-AdaIN/tree/master

[3] H. Y. Chen, I. S. Fang, C. M. Cheng, and W. C. Chiu, "Self-contained stylization via steganography for reverse and serial style transfer," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[4] M. Tomei, M. Cornia, L. Baraldi, and R. Cucchiara, "Art2real: Unfolding the reality of artworks via semantically-aware image-to-image translation," 2019.

[5] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2 - A large-scale benchmark for instance-level recognition and retrieval," *CoRR*, vol. abs/2004.01804, 2020. [Online]. Available: https://github.com/cvdfoundation/google-landmark

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

[7] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2021.

[8] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," 2023.

[9] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

VII. Appendix