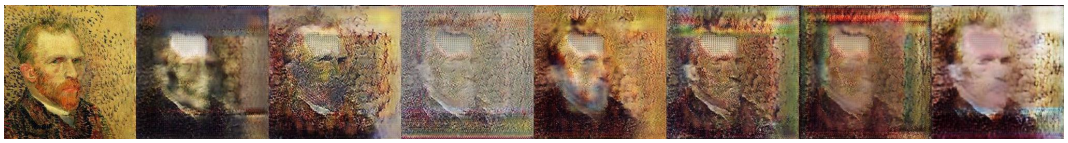

Generative Adversarial Networks for Multi-Domain Art Creation

Allan Bishop
adb262@cornell.edu

Anthony Yang
axy2@cornell.edu

Noah Rush
njr86@cornell.edu

Abstract

Creating art has always been a very human task due to the complexity and subjectivity of process. With the advent of deep learning, specifically Generative Adversarial Networks, AI has begun to create original art. There exists approaches for transferring the style of one picture to the content of another, but in this project we attempt to capture the style over a group of images as defined by their genre. In this paper, we explore two GAN architectures, CycleGAN, and StarGAN. Our objective is to create a model capable of handling translation across multiple genre domains, i.e. converting an Impressionist painting to an Expressionism painting, or generating an image in different genres from one photograph. Because of the domain limitations of CycleGAN, we focused more on implementing StarGAN, a relatively novel and scalable approach that can perform image-to-image translation for multiple domains using only a single model. Using StarGan, we were able to generate new images across several genres of painting and real images, and evaluate the success of our model over various domains using the Frechet-Inception Distance.

1 Introduction

The original motivation stemmed from our interest in working with image based data, especially art and paintings. After doing a little research, we found a Kaggle competition [11] that matched our interests and was of suitable academic complexity. The Kaggle competition aims to generate Monet paintings using the CycleGAN architecture, but we decided to expand the domain to encompass multiple styles, generating artwork in a variety of different styles from real images and other art. In this context, we denote domain to encompass a specific style of art, i.e. impressionism or cubism. However, CycleGANS are mainly designed to translate between two domains, and introducing new domains require training 2 new discriminators and 2 new generators. As a result, we look to implement the StarGAN architecture.[5] Ultimately, this project aims to create an application that can take an image and a chosen target style, and output the same image transformed under the guise of the chosen target style.

2 Background

Traditionally, training image to image models required a dataset with paired examples. However, with the advent of GANs, this requirement was no longer necessary. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated).

CycleGAN CycleGANs are composed of two sets of a generator-discriminator pair. Let's assume that our goal is to turn a real image into a Monet-style painting. The first generator is fed the real life image and then outputs a fake Monet version. Then D1 (Discriminator 1) is then fed real Monet paintings and fake Monet images that G1 (generator 1) outputs and tries to correctly tell the difference. Next, G2 is fed the fake monet images of G1 and tries to output the real life image. D2 will then try to discriminate between the generated real life images and actual real life images.

Applied Machine Learning Project Report. Fall 2020.

The above example shows the difficulty of scaling up the cycleGan to new domains. Each generator and discriminator have a specific target domain and corresponding generators and discriminators to work with. Adding a new domain, say real pictures to Van Gogh, would require training a generator of Van Goghs(VGs) from real images, a discriminator between VGs and generated VGs, a generator of real photos from VGs, and a discriminator between photos and regenerated photos.

StarGAN [5] introduced in 2018, presents a unified method for doing multi-domain image translation, which is what we will mainly explore in this paper.

3 Method

We train a single generator *G*, that when given an input image x and a target domain c, outputs the c domain version of x. Furthermore, we train a discriminator that is inputted images from a pool of real and fake images and must both distinguish whether the input is generated or not and must be able to output the domain classification of the real images.

Adversarial Loss The adversarial loss function is utilized by both the generator and the discriminator to ensure that generated images are indistinguishable from real images. The key is that the generator is trying to minimize this function while the discriminator is trying to maximize. As a result, this adversarial dynamic between the generator and the discriminator allows for constant improvement and competition.

Domain Classification Loss The discriminator, on top of distinguishing between real and fake images, must also be able to correctly classify the real images to the correct domain.

Reconstruction Loss By minimizing the adversarial and classification losses,*G*is trained to generate images that are identifiable to its correct target domain. The reconstruction loss function runs the generated image $G(x, c)$, through the generator back to its original domain $G(G(x, c), c')$. The goal of the reconstruction loss is to preserve the content of the original image.

Objective The final objective function, written in terms of generator *G* and discriminator *D*

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^r_{cls} \tag{1}$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^f_{cls} + \lambda_{rec}\mathcal{L}_{rec} \tag{2}$$

where λ_{cls} and λ_{rec} are both hyper-parameters indicating how important the domain classification and reconstruction loss are. [See appendix for more details on the loss functions.]

Network Architecture The network architecture of StarGAN is composed of a generator and discriminator. The generator network consists of two convolutional layers with the stride size of two for downsampling, six residual blocks, and two transposed convolutional layers with the stride size of two for upsampling. The discriminator network consists of five hidden convolutional layers with a stride size of two. [see appendix figs 7 and 8 for picture of network]

4 Experiments

We started with the art dataset offered by Kaggle in the "Painter by Numbers" competition[9]. We filtered out the paintings tagged by portraits and split them by their genre tag. The genres contained a variable amount of images but we tried to use genres that had at least 200-1000 examples. We combined the portrait dataset with images from the "images of groups" dataset[10], which contained pictures of groups of people. We split out 2000 images for our test/dev set. Because StarGAN is able to learn multi-domain mappings, we were able to train the model on all of the genres at the same time. The input images were all 256x256 RGB images. Initially, we trained the model on the images with a random crop to align the data with the correct input dimensions. StarGAN was trained with a learning rate of 0.0001 for both the Generator and Discriminator. Each StarGAN was trained to at least the 60,000th iteration.

As a result of poor random cropping, we often missed the face in our images. Figure 4 demonstrates the mappings of some poorly cropped images. Although the face is not the all-determining factor of the genre, as a consistent feature across all images, it can limit the learning domain of the generator. In our initial test, we also allowed the model to try and map images of all domains to our selected genres. However, this was a mistake because instead of keeping track of the domain source, it would lump all non-specified genres into one class (i.e. abstract = inked - washed = renaissance) Knowing these two things, we decided to use PyPi Face-Recognition [3] to help identify and crop the faces out of the portrait and groups of people datasets. PyPi Face-recognition did a fairly good job of identifying faces in art, achieving success over 50% of the time. We also limited

the dataset to only the images in our new target domains (Expressionism, Romanticism, High Renaissance, Impressionism, Post-Impressionism, Pointillism and Real(Faces)). This ended up containing 5216 images. With this new dataset we retrained our model and plotted our results, and unsurprisingly, got better results both subjectively and with our FID score.

We used both qualitative and quantitative evaluation, with Frechet Inception Distance (FID) functioning as our quantitative metric. Introduced an [4], FID is a metric that uses a pre-trained neural network to pull intermediate features out of images. 'Inception' in FID refers to the inception.pb neural network that is used. The network works on both the resulting and input images and pulls the same feature set. It allows us to measure the quality of our generated images by comparing low level features. Lower FID score has been shown to correspond with human judgment and image disturbances [4] like salt and pepper noise, blur and swirling[4]. Thus, we take a lower FID score to mean the model has learned a better mapping between classes, and outputted better results.. We plotted iterations vs. FID score to trace the model's generative abilities over time. Unsurprisingly, our cleaned dataset (face crops and limited labels) had consistently lower FID scores.

We also note here that for the original Kaggle competition, Kaggle uses a slightly different metric to score results. Memorization Informed Frèchet Inception Distance (MIFID) takes Cosine Distance (d) between the source set and resulting images and punishes those with extremely low d. MIFID allows for FID similarity analysis while also penalizing learning the images, not the mappings. Kaggle implements this to punish submissions that are literally Monet or slightly changed Monet images, so this is not a concern for us.

The face cropped images yielded results that suggested the model was getting better over time, improving [on average] over iterations. However, the randomly cropped images mostly suggested that the model was best trained after an early iteration. Figures 6 and 7 present the iteration-FID plot for each training set. We were able to achieve our best FID scores of 146 for Romanticism with the FaceCrop dataset while we could not beat 250 with our randomly cropped data. This demonstrates how poor data led to mappings being more obscure and difficult to learn. Another advantage of the StarGAN process is the ability to compare separate performance for various genres at once. For example, looking at the results, we can see optimal training points for Pointillism and Impressionism, and if they are different generate images from the point in the model where they are optimal.

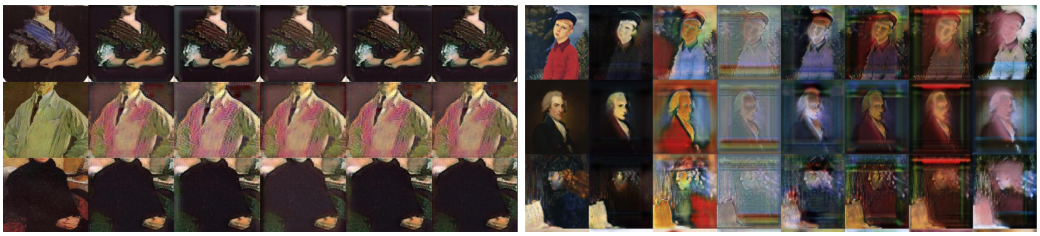

Figure 1: Poor Random Cropping Vs. Using Face Detection

5 Evaluation

5.1 Quantitative

In Figure 2, we look at the FID scores for the first faulty experiment, over epochs. Each FID score is calculated between 16 sample generations and the original paintings in that genre. In figure 3, we see the scores for the training set over epochs with the corrected data set. Note that the genres in both experiments (High Renaissance, Impressionism and Expressionism) reach lower FID scores in the second experiment. In figure 4, we see the test results over epochs, and note that the FID score is even lower.

5.2 Qualitative

Using both StarGAN and CycleGAN, we were able to generate some compelling new art images and some compelling art to real world images. With CycleGAN, we tried generating art from various art from various genres, and got varying results depending on the consistency of the source genre style. For example figure 5 shows a clear result for generating a pointillist painting from a photograph, compared to trying a generating pop art, a widely variant art style.

For StarGAN, we could generate lots of results across genres, and compare success without training new models. The results varied widely, both over testing epoch, and from domain source to domain target(See appendix fig. 9), and some images in out dataset were just better for transformation than others. We were able to generate some interesting results. In figure 8, we see one of Van Gogh's self-portraits as the input image, and to transformations, one to expressionism and one to a real image. Note the bold features in the expressionism version, and the way the real version, though blurry and distorted begins to resemble more of an old photograph. See appendix figs. 10,11 and 12for more results.

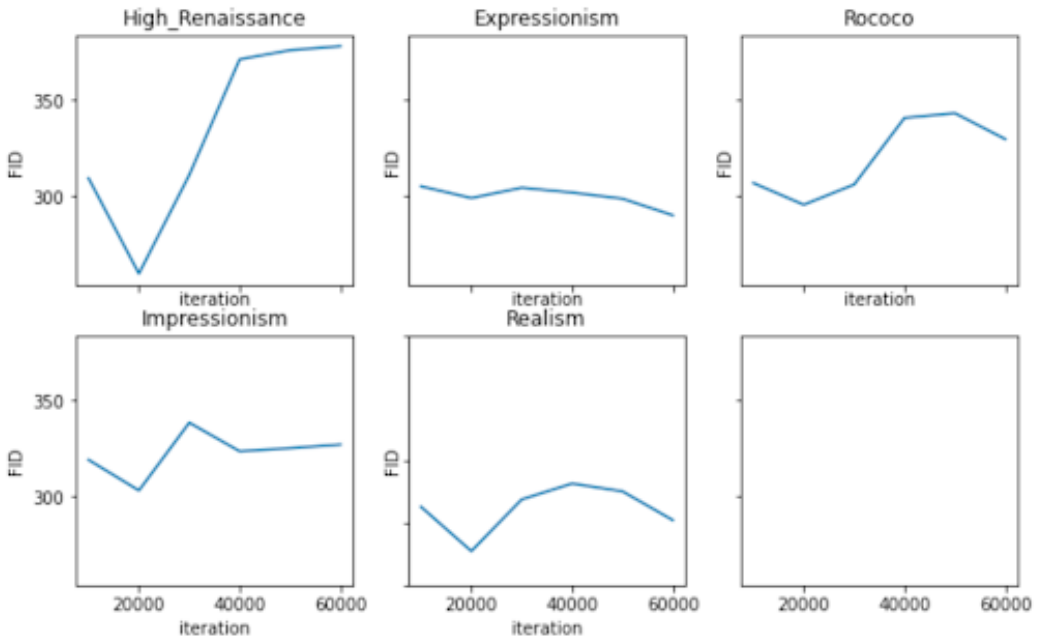

Figure 2: FID Scores over epoch on random-cropped dataset

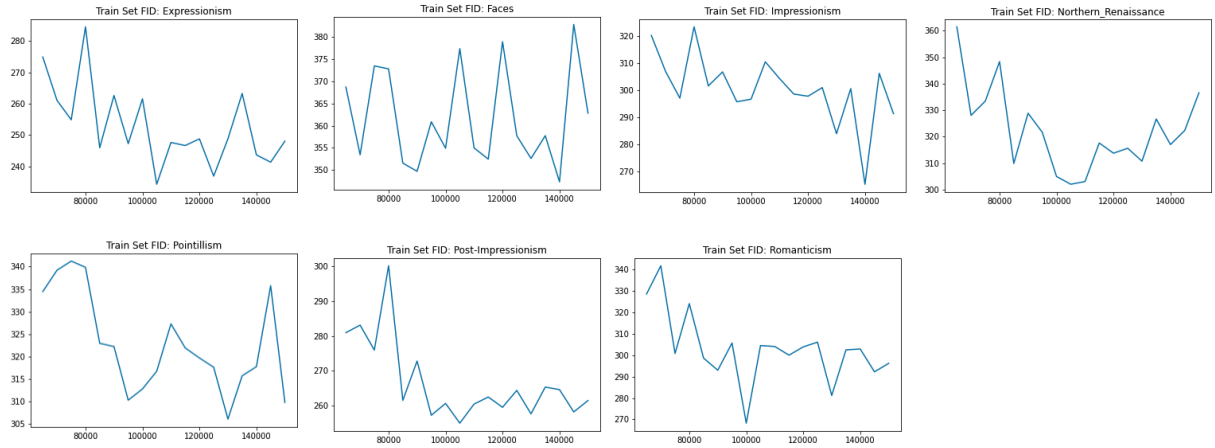

Figure 3: FID Scores over epoch on face-cropped dataset, unlabeled domain removed

6 Discussion and Prior Work

Although we do see a fairly significant drop in FID from train to test, we can attribute this to the sample size of the FID data. The training FID's were derived from sample translations that StarGAN would output at intermediate iterations. These translations were small in number (48) and were compared to our entire control set to get that iteration's FID. Our test FID's were derived from the saved model's translation of 2,000 test images, thus allowing a more concrete comparison to the control. We notice the heavy spikes in figures 3 and 4. We assume these spikes to be related to the cyclical nature of GAN's and the aversion to converge that these models hold [4]. It is common for GAN's to get better, worse and subsequently better as they learn the feature mappings.

We note more impressive results in driving down the FID score for Expressionism and Post-Impressionism, and recognize that these are styles with more flexibility in construct, having more abstract styles than the other genres and also recognize that in our qualitative analysis, these are styles that we thought generated more interesting and consistent artwork.

It would also be interesting to note that the various size constraints our dataset had, even though they were all por-

Figure 4: FID Scores over epoch on face-cropped dataset, unlabeled domain removed

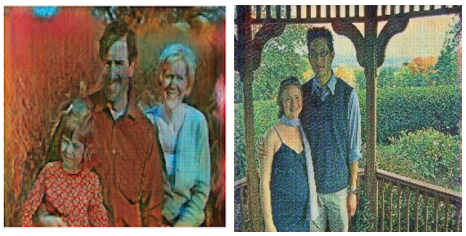

Figure 5: CycleGAN: Certain genres yield more distinctive results, image on right is clearly pointillism

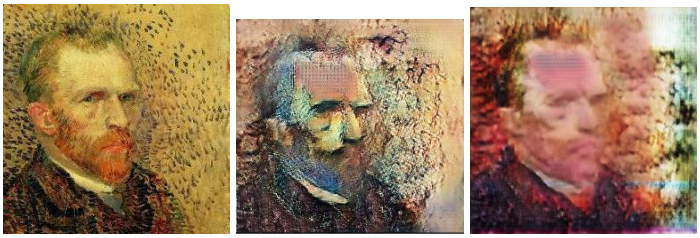

Figure 6: StarGAN: Van Gogh self-portrait transformed to Expressionism and Photo

traits, we working with a dataset initially cropped to 256 in one dimension. In the Stargan paper, they resulted unexpected success in the generator learning human faces in each domain, over multiple domains. This was helped by the fact that they had a consistent dataset with all the faces the same size. Perhaps if we had a dataset more centrally focused on faces we could have had more consistent results.

7 Conclusion

We can conclude that StarGAN yields an effective image-to-image multi-domain mapping. It has shown it's ability to translate images across domains with comparable quality and accuracy to CycleGAN. StarGAN's primary advantage over CycleGAN is the multi-domain nature of training and it's efficient comprehension of each genres mappings. It also provides a way to simultaneously compare the effectiveness of domain transfers. As a result, StarGAN provides an significant advantage in terms of scalability and efficiency. Although StarGAN does not converge, as the iterations progressed we were able to drive FID down and achieve improved translations. In the future, we would like to learn about how utilize methods to improve GAN convergence, reduce distortion in our art generation, and finally, experiment with other artistic scenarios where multi-domain translation could produce interesting results. We hope that our work provided a refreshing and novel effort in automating a traditionally human-dominated field.

8 Appendix

Adversarial Loss

$$\mathcal{L}_{adv} = \mathbb{E}_x \left[\log D_{src}(x)\right] + \mathbb{E}_{x,c}\left[\log\left(1 - D_{src}(G(x,c))\right)\right] \tag{3}$$

where *G* generates image *G(x,c)* and D tries to classify the image as fake or real.

Domain Classification Loss

$$\mathcal{L}_{cls}^{r} = \mathbb{E}_{x,c'}\left[-\log D_{cls}\left(c' \mid x\right)\right] \tag{4}$$

where $D_{cls}\left(c' \mid x\right)$ is the probability distribution of the domain labels created by *D*.

Reconstruction Loss

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'}\left[\left\|x - G\left(G(x,c), c'\right)\right\|_1\right] \tag{5}$$

where *G* takes in the generated image and a original domain label *c'* and outputs the original image *x*.

Part	Input → Output Shape	Layer Information
Down-sampling	$(h, w, 3 + n_c) \rightarrow (h, w, 64)$	CONV-(N64, K7x7, S1, P3), IN, ReLU
	$(h, w, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	CONV-(N256, K4x4, S2, P1), IN, ReLU
Bottleneck	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
Up-sampling	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	DECONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 64)$	DECONV-(N64, K4x4, S2, P1), IN, ReLU
	$(h, w, 64) \rightarrow (h, w, 3)$	CONV-(N3, K7x7, S1, P3), Tanh

Figure 7: Generator Network Architecture

Layer	Input → Output Shape	Layer Information
Input Layer	$(h, w, 3) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{32}, \frac{w}{32}, 1024)$	CONV-(N1024, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{32}, \frac{w}{32}, 1024) \rightarrow (\frac{h}{64}, \frac{w}{64}, 2048)$	CONV-(N2048, K4x4, S2, P1), Leaky ReLU
Output Layer (D_{src})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (\frac{h}{64}, \frac{w}{64}, 1)$	CONV-(N1, K3x3, S1, P1)
Output Layer (D_{cls})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (1, 1, n_d)$	CONV-(N(n_d), K$\frac{h}{64}$x$\frac{w}{64}$, S1, P0)

Figure 8: Discriminator Network Architecture

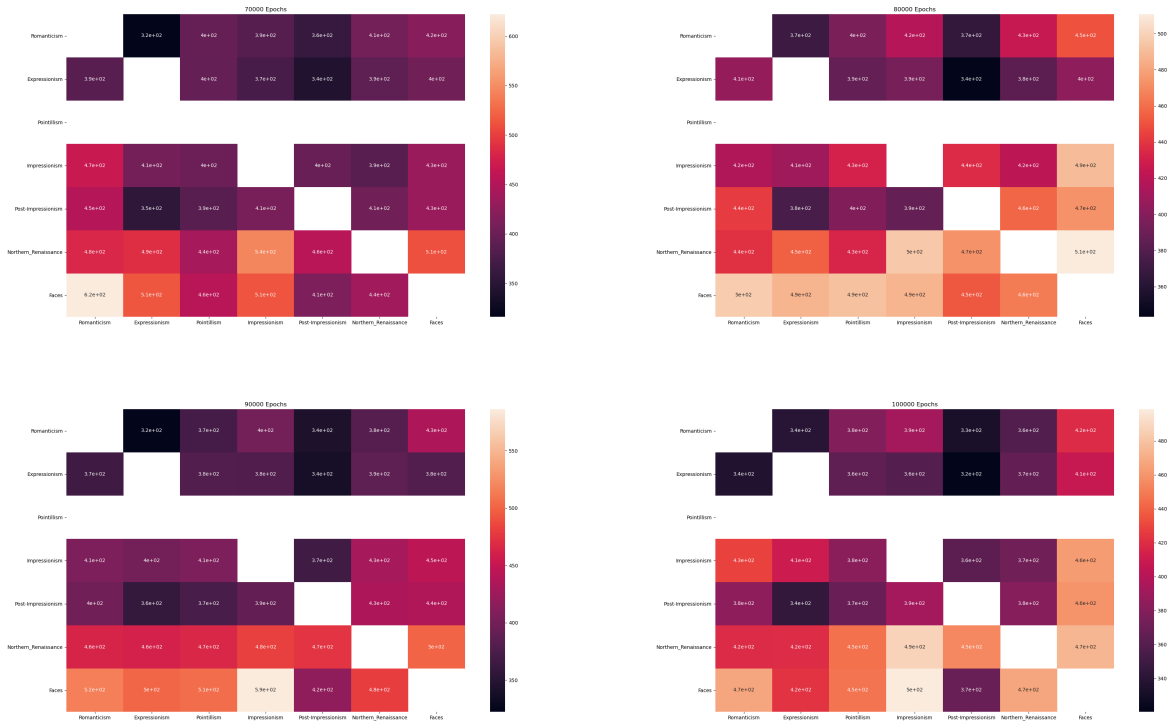

Figure 9: Domain translation evaluated between different source genres and different target genres at various epochs. Data taken from 48 sample images across 6 domains (not including pointillism). Value is the FID.

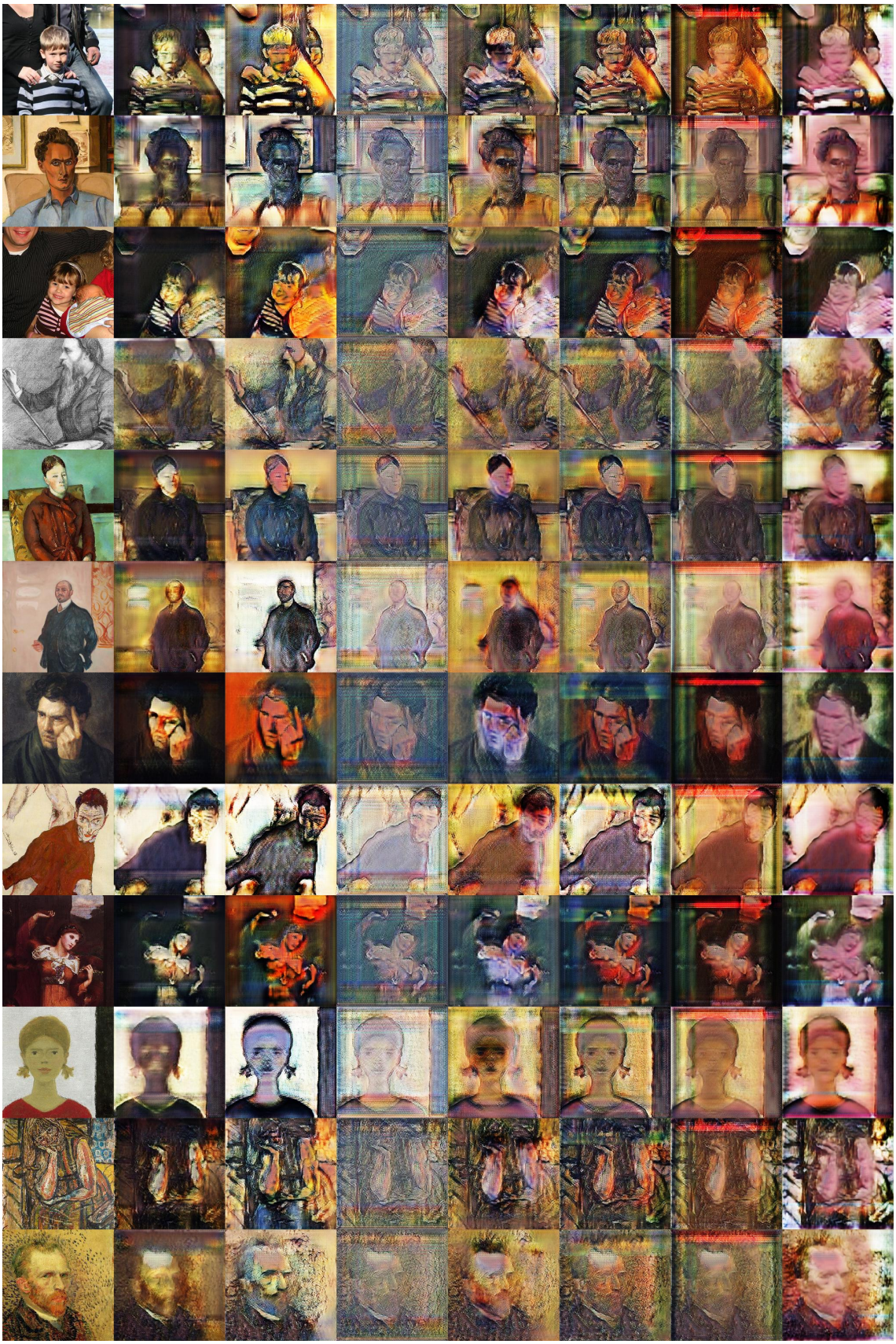

Figure 10: Art translations from the generator trained for 100000 epochs. Original image is on the left. Genres to the right of that are as follows: Romanticism, Expressionism, Pointillism, Impressionism, Post-Impressionism, High-Renaissance, Real

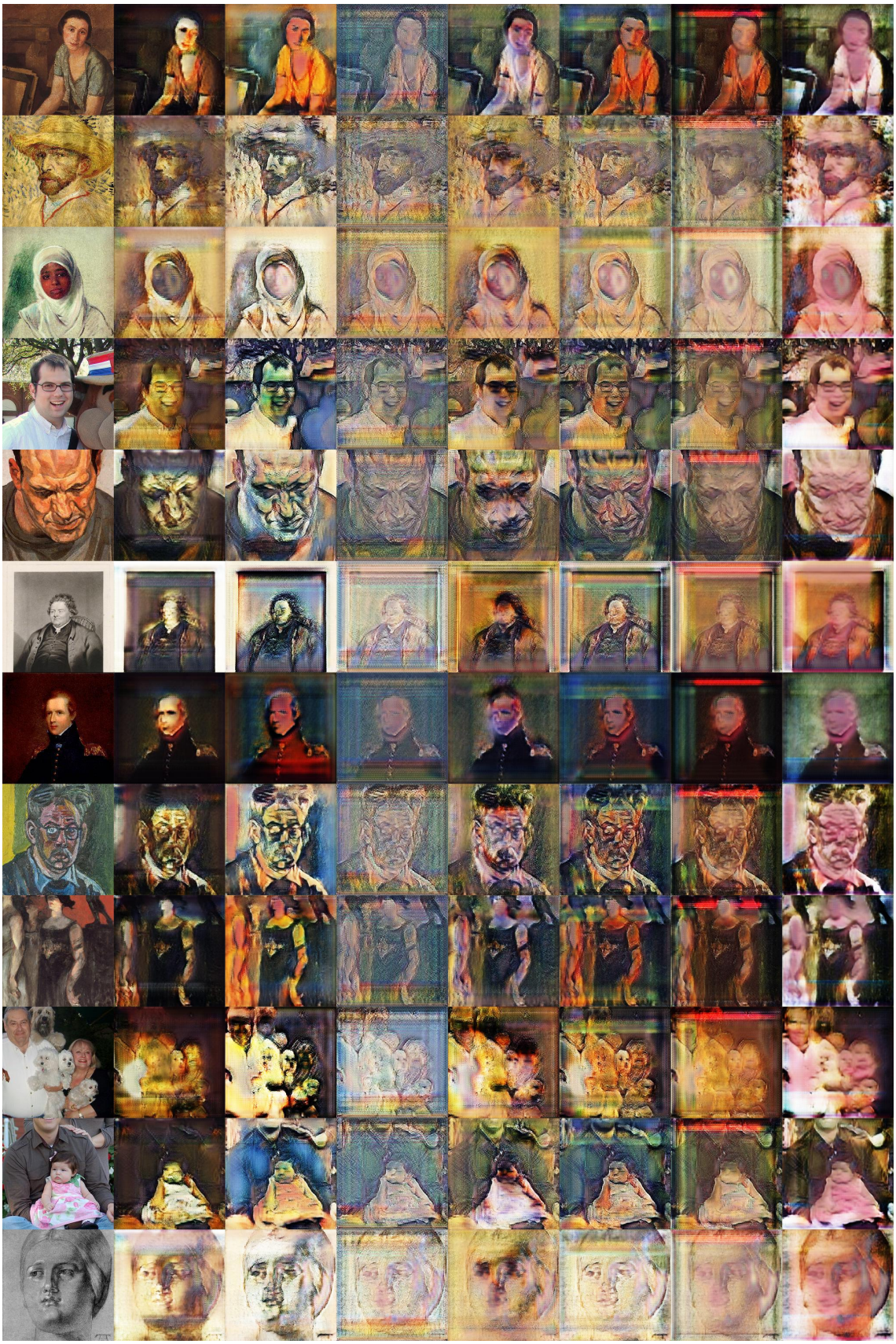

Figure 11: Art translations from the generator trained for 100000 epochs. Original image is on the left. Genres to the right of that are as follows: Romanticism, Expressionism, Pointillism, Impressionism, Post-Impressionism, High-Renaissance, Real

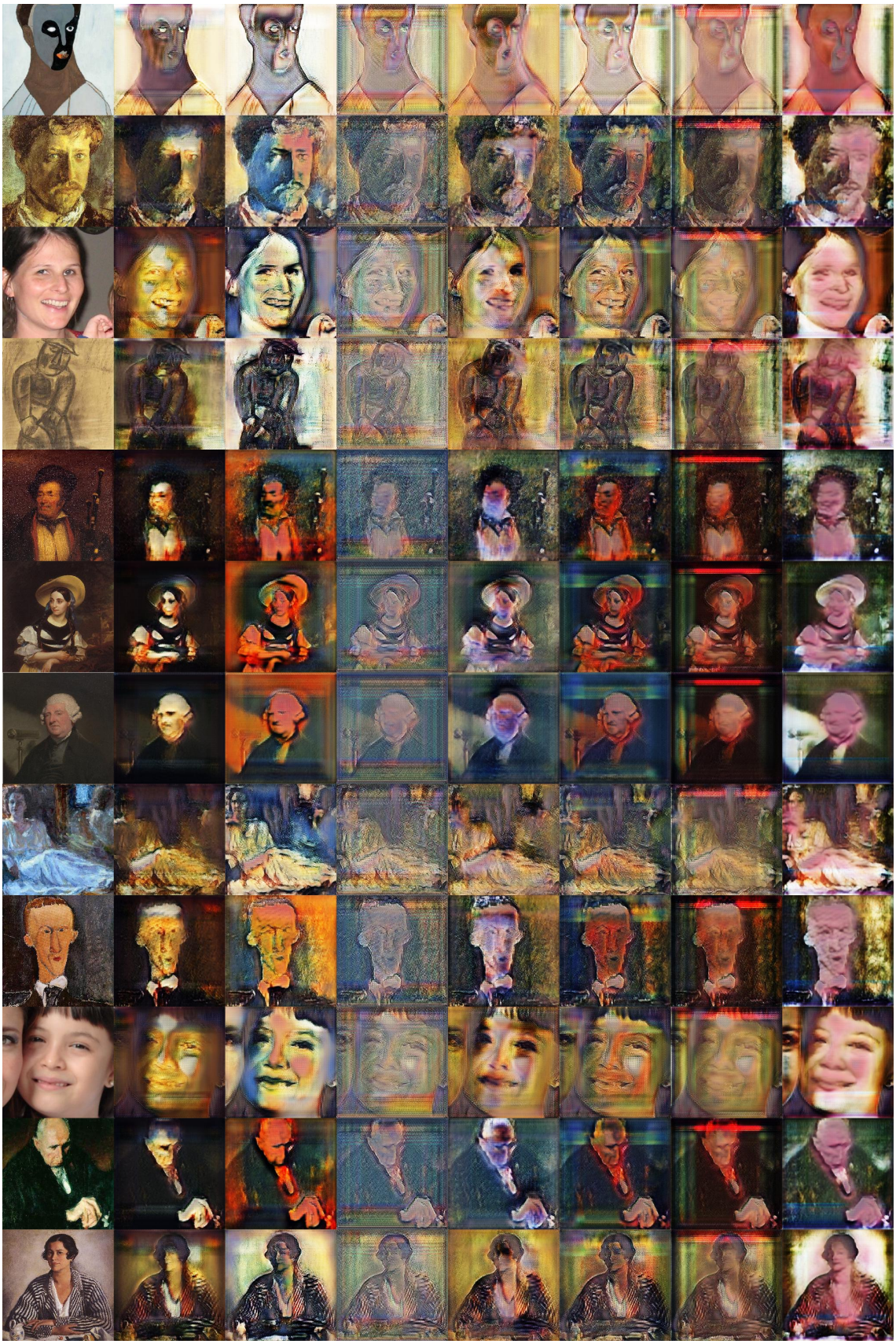

Figure 12: Art translations from the generator trained for 100000 epochs. Original image is on the left. Genres to the right of that are as follows: Romanticism, Expressionism, Pointillism, Impressionism, Post-Impressionism, High-Renaissance, Real

References

[1] Choi, Y. StarGAN - Official PyTorch Implementation, - https://github.com/yunjey/starGAN

[2] Seitzer, M. pytorch-fid: FID Score for PyTorch, - https://github.com/mseitzer/pytorch-fid

[3] Geitgey, A. Face Recognition - https://github.com/ageitgey/face_recognition

[4] Martin Heusel and Hubert Ramsauer and Thomas Unterthiner and Bernhard Nessler and Günter Klambauer and Sepp Hochreiter (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Nash EquilibriumCoRR, abs/1706.08500.

[5] Yunjey Choi and Min-Je Choi and Munyoung Kim and Jung-Woo Ha and Sunghun Kim and Jaegul Choo (2017). StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image TranslationCoRR, abs/1711.09020.

[6] TensorFlow CycleGAN - https://www.tensorflow.org/tutorials/generative/cyclegan

[7] Jang, Amy. CycleGAN Tutorial -https://www.kaggle.com/amyjang/monet-cyclegan-tutorial

[8] Tensorflow Normalizations - https://www.tensorflow.org/addons/tutorials/layers_normalizations

[9] Kaggle - Painter By Number - https://www.kaggle.com/c/painter-by-number

[10] A. Gallagher, T. Chen, Understanding Groups of Images of People, IEEE Conference on Computer Vision and Pattern Recognition, 2009. Available here : http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html

[11] Kaggle - I'm Something of a Painter Myself - https://www.kaggle.com/c/gan-getting-started