GANtor Arts Center

Avoy Datta
Department of EE
Stanford University
avoy.datta@stanford.edu

Michael Cai
Department of CS
Stanford University
mcai88@stanford.edu

Figure 1: Sample outputs from the GANtor-v2 Stage-II generator interspersed with real artwork from the Wikiart dataset. Can you spot real from fake? Scroll down to the appendix and see Figure 20 for the answer!

Abstract

*In this work, we have constructed and analyzed a novel Stacked Generative Adversarial Network for the task of class-specific art generation. Our model combines the specialized conditional image generation and discrimination architecture of **ArtGAN** with the **StackGAN** architecture's ability to generate high quality high-dimensional images in order to generate 256x256 images conditioned on different classes. We trained our model on samples from the Wikiart dataset labeled based on style and genre. Our model results are evaluated via the Frchet Inception Distance (FID), Inception Score (IS), Inception Classification (IC), and human evaluation. We found that our two-stage network encountered training instability after about 50 epochs of training for Stage-II, but still outperformed our baseline on the basis of Inception Score.*

1. Introduction

Generative Adversarial Networks (GANs) have an impressive track record for generating very realistic-looking artwork. The first architectures used to produce GAN-generated visual art [8] were trained on one particular genre or class of art and then applied to produce novel samples from the ground truth distribution. In the last couple of years, there has been surging interest in generating art **conditioned** on class or genre labels, with the goal of enhancing the degree of control over the type of artwork produced by generative models. Training a GAN on a large dataset of multiple classes and generating art conditioned on a class, as opposed to training and generation for a single class, allows the generator to learn correlated features across different classes. Being able to tune the art produced to one's liking also adds to the set of interesting applications for these networks. In this project, we will discuss our work on **GANtor-v2**, a 2-stage stacked GAN trained to generate 256x256 works of art conditioned on different *genres*.

2. Related Work

State of the art performance for art generation was first achieved in 2017 by Wei, et. al. with their **ArtGAN** architecture [11]. ArtGAN extends the original GAN template first proposed by Goodfellow, et. al. [3] by appending a conditioning class vector to the Gaussian noise vector passed into the generator to produce 128x128 outputs from different classes. Then their discriminator, along with distinguishing between real and fake works, produces

both a classification score for the genre of passed-in images and a reconstructed image. Finally the loss function sums the cross entropy losses and reconstruction loss, which are backpropagated through the network. The ArtGAN authors quantitatively evaluated their network using the log-likelihood metric, and achieved a score of 2564 +/- 67. In addition, the authors also propose numerous variants incorporating advanced architectural features such as autoencoders to further improve model performance.

GANGogh [5] is a similar project that tackles the problem of Art generation using a slightly different approach. It incorporates the Wasserstein distance from **Wasserstein-GANs** [1] into a modified **AC-GAN**[7] to obtain comparable results to ArtGAN. One of the extensions proposed in GANGogh, but not made public at the time of this writing, was the use of a multi-stage GAN to enforce differentiability between outputs.

The use of multistage GANs for conditional image generation was explored by Zhang, et. al. with their **StackGAN** architecture, which generated images conditioned on text captions using a two-stage GAN [12]. To generate images, StackGAN appends the text embeddings for each caption to a randomly generated noise vector that is used by the Stage-I generator/discriminator to produce 64x64 images which are then used as input to the Stage-II generator/discriminator, ultimately resulting in 256x256 image outputs. The goal of Stage-I is to produce low-dimensional images that maximize differentiability, while Stage-II serves as a platform to maximize saliency and diversity [12]. This idea is explored further in our work.

3. Conditional Art Generation with Stacked Adversarial Training

While ArtGAN is currently the state of the art in this field, we believe the multi-tiered GAN architecture proposed in **StackGANs** [12] can elevate differentiability between the art produced for different genres/labels, an extension first proposed by the authors in [5]. The stage-I GAN could serve as a means of enforcing differentiability between genres, creating low-resolution images that could be fed into the stage-II GAN alongside the global context vector (containing information from the desired label) which would then produce the art. This could help attain both of goals (2) and (3) above, by preventing different input noise vectors from mapping to similar images once training has concluded. The loss functions of both discriminators would need to take into account not just the classification between 'real' and 'fake' images, but also the classification label of the generated (fake) images conditioned on the context vectors appended to the input.

In summary, our model should optimize the following:

- **Differentiability** of outputs conditioned on input class
- **Diversity** of generated samples **across all classes**
- **Saliency** of generated samples

A fully-trained model has the following I/O signatures:

Inputs:

- A context-vector (**one-hot**) indicating the style/genre of art we would like to generate.
- A noise vector sampled from a standard normal distribution
- $(64)^2$ images during Stage-I training to calculate the Discriminator-I loss
- $(256)^2$ images during Stage-II training to calculate the Discriminator-II loss

Outputs:

- $(256)^2$ images from Stage-II with each image's class label *ideally* matching the corresponding desired class passed in through the context vector.

Note: Because our main motivation for using a two-stage adversarial network is to generate high-resolution, realistic samples, the Stage-I outputs are not considered the final outputs of the model. We evaluate the Stage-I outputs all the same to get a better understanding oh how generated image quality changes from Stage-I to Stage-II.

4. Methodology

4.1. Baseline

Neither of the papers attempting conditional art generation ([11], [5]) reported standard quantitative metrics to evaluate their results. [11] reported a loglikelihood of generated results instead. We wanted to evaluate our model with more standard metrics than loglikelihood, and due to the lack of availability of a trained ArtGAN model in Pytorch (the library we used for development), and computational constraints preventing us from training training the ArtGAN model from scratch, we decided to implement our own baseline architecture based on a 2-layer stacked GAN, modifying the loss functions to align with those used in [12].

4.1.1 Architecture

As our baseline model, we propose **GANtor-v1**(figure 3), a 2-Stage GAN that generates art conditioned on artist styles. The individual generator and discriminator architectures adapted from the preexisting Stack-GAN model built by

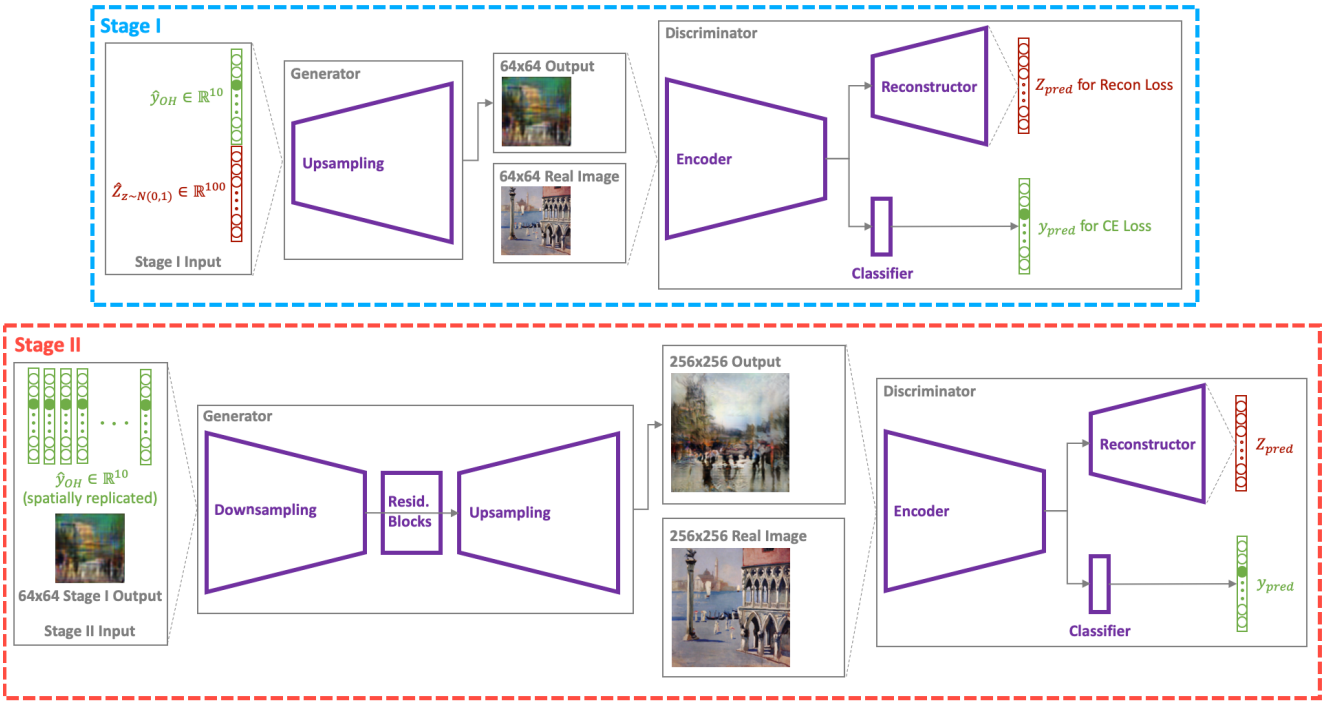

Figure 2: GANtor-v2 Architecture

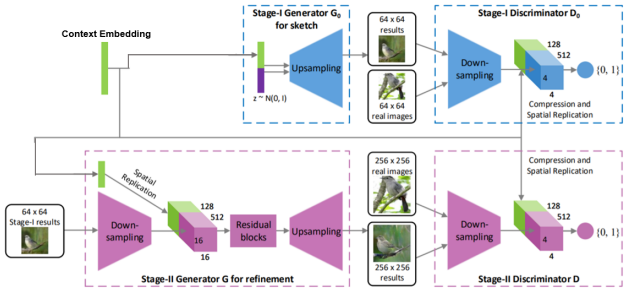

Figure 3: GANtor-v1 (Baseline) Architecture. Figure modified from the one presented in [12]

Zhang et. al. [12] that outputs images conditioned on embeddings generated from text captions. We built our baseline model based on their architecture to directly accept a one-hot style vector representing a style category (i.e. index 0 for Abstract Expressionism, 11 for High Renaissance, etc.) which we append to the noise inputs of each stage of our GAN.

The Stage I Generator accepts a 100 dimensional noise vector sampled from a standard normal distribution and outputs a 64x64 image. When training Stage I, this image is then compared to a 64x64 crop of a real artwork from the same category by the Stage I Discriminator. Once Stage I has been fully trained, we then train Stage II by generating a 64x64 image using the fully-trained Stage I Generator, then feeding this image into Stage II conditioned on the same style vector in order to output a 256x256 generated image. Finally, this output is again compared to a 256x256 crop of a real art piece by the Stage II Discriminator.

4.1.2 Loss Functions

The GANtor-v1 uses a modified variant of the losses used in [12]. The Stage-I GAN trains by alternatively maximizing L_{D1} and minimizing L_{G1}.

$$L_{G1} = \mathbb{E}_{(I_1,t)\sim p_{data}}[\log D_1(I_1, c_t)] + \mathbb{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_1(z, c_0), \phi_t))]$$
$$L_{D1} = \mathbb{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_1(G_1(z, c_0), \phi_t)]$$

The Stage-II losses are conditioned on the Stage-I outputs $s_1 = G_1(z, c_0)$:

$$L_{G2} = \mathbb{E}_{(I_2,t)\sim p_{data}}[logD_2(I_2, c_t)] + \mathbb{E}_{s_1\sim p_{G1}, t\sim p_{data}}[log(1 - D_2(G_2(s_1, c_0), \phi_t))]$$
$$L_{D2} = \mathbb{E}_{s_1\sim p_{G1}, t\sim p_{data}}[log(1 - D_2(G_2(s_1, c_0), \phi_t)]$$

4.2. GANtor-v2

4.2.1 Architecture

Stage-I of the GANtor-v2(figure 2) was inspired by the architecture in ArtGAN [11]. It improves over GANtor-v1 by using a convolutional autoencoder inside the discriminator, and an L2 pixelwise reconstruction loss between the inputs and outputs of this autoencoder to stabilize Generator training.

The Generator uses a series of upsampling blocks and convolutional layers to map a concatenated input of a 10-dimensional context embedding (a one-hot vector) and a 100-dimensional noise vector to a 64x64 image. The discriminator takes in both this 64-squared dimensional 'fake' image and a 'real' image of the same dimensions to compute the Stage-I discriminator loss. The outputs of the Stage-I are fed into the **Stage-I Stage-II** Generator, which also takes in a *spatially replicated* copy of the *same* context embedding used to generate the Stage-I result. The Stage-II Generator outputs *two* images - one a 256x256 sample that is the final output of the network, and the other a downsampled version of the final output that is used to compute the Stage-II discriminator loss. A more detailed description of the GANtor-v2 architecture is provided in figure in the appendix.

4.2.2 Loss Functions

Both stages of GANtor-v2 use the same loss functions used in [11]. For Stage-II, we modified the dimensions of the embeddings used in the stage-II reconstruction loss to account for the larger dimensions of images produced (256x256 as opposed to 128x128 in ArtGAN), but the objectives were otherwise unchanged. The ArtGAN Loss discriminator loss seeks to minimize a superposition of the cross entropy between real and fake labels and a cross entropy loss for whether the correct class was outputted.

The generator loss seeks to *maximize* the cross entropy loss for the correct class. To stabilize training, the L2 pixelwise reconstruction loss between the reconstructed images at the discriminator and discriminator inputs is also added to the Generator loss. The equations for $\mathcal{L}_D$, $\mathcal{L}_G$, and $\mathcal{L}_{L2}$ are given below:

$$\mathcal{L}_{L2} = \mathbb{E}_{x_r \sim p_{data}}[||Dec(Enc(x_r)) - x_r||_2^2]$$

$$\begin{aligned}\mathcal{L}_D = &-\mathbb{E}_{(x_r,k)\sim p_{data}}[\log p(y_i|x_r, i=k) \\ &+ \log(1 - p(y_i|x_r, i \neq k))] \\ &- \mathbb{E}_{\hat{z}\sim p_{noise}, \hat{k}\sim K}[\log(1 - p(y_i|G(\hat{z}, \hat{y}_{\hat{k}}), i < K+1)) \\ &+ \log p(y_i|G(\hat{z}, \hat{y}_{\hat{k}}), i = K+1)]\end{aligned}$$

$$\mathcal{L}_G = -\mathbb{E}_{\hat{z}\sim p_{noise}, \hat{k}\sim K}[\log p(y_i|G(\hat{z}, \hat{y}_{\hat{k}}), i \neq \hat{k}))] + \mathcal{L}_{L2}$$

Due to computational and resource constraints, we were only able to train and evaluate the GANtor-v2 on *genre* labels. From our experiments fine-tuning the Inception-v3 on the Wikiart dataset, we found that *genre* yielded the highest top-1 classification accuracy during validation (66% after 5 epochs). Along with the fact that *genre* is qualitatively easier to distinguish than the other two categories provided in the dataset (*style* and *artist*), we felt this would the most optimal category to evaluate our model on.

To train GANtor-v2 we use the same training algorithm used in ArtGAN. Pseudocode for this algorithm is provided in figure 13.

4.3. Evaluation

Instead of evaluating performance with a single metric, we use three separate metrics to evaluate our model:

- **Inception Score (I.S.)**: The Inception Score (I.S.) proposed in [9] uses a large set of generated GAN outputs to compute underlying global statistics of the generated distribution, prioritizing both *saliency* and *diversity* in its scoring. It computes the **expected Kullback-Liebler(KL) Divergence**[6] between the generated sample distribution conditioned on each class label, and the marginal sample distribution. The sharper the former distribution (i.e. samples are easy to distinguish for each class) and the more uniform the latter (i.e. large diversity in overall class labels generated) the higher the I.S. will be.

  $$\text{IS(G)} = \exp(\mathbb{E}_{x\sim p_g} D_{KL}(p(y|x)||p(y)))$$

  The I.S. is limited by what the Inception classifier can detect, and struggles when evaluating GANs trained on data that is difficult to classify[2]. This is particularly the case when the Inception v3 [10] network used in evaluating this metric has not achieved high rates of accuracy even after pretraining on the ground truth dataset. This aligns with the line of reasoning provided for the lack of reporting of an inception score in the ArtGAN paper [11], where the authors instead reported the final log-likelihood of results produced by their network. We therefore use two additional metrics for evaluation.

- **Top-1 classification accuracy:** on the generated samples: The top-1 accuracy is the accuracy with which the Inception-v3 is able to predict the 'correct' class of a generated sample, where the 'correct' class is the class indicated through the context vector(2) passed into the Generator as input. This metric is used as a measure of how *correctly differentiable* our samples are conditioned on class labels. If the predicted class label (from Inception) of a generated sample matches the class that was passed in through the context vector

Figure 4: GANtor-v2 Stage-II outputs by genre. Real sample images are given in the red box

then that sample will contribute positively towards this metric.

- **Frechet Inception Distance (F.I.D.)**: The **Frechet Inception Distance**[4], or F.I.D., is a measure of how different two different sets of image data are, given by the difference in the activations of the Pool3 layer of the Inception-V3 network. Specifically, given activations $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$, the FID is given by:

$$\text{FID} = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

  The F.I.D. is often considered a more comprehensive metric for measuring GAN performance because it comares outputs to real samples rather than evaluating them in a vaccum, and because it is more robust to noise in the data compared to the I.S.

All three of our metrics were evaluated using an **Inception v-3**[10] network pretrained on the ImageNet dataset and fine-tuned on the Wikiart dataset for 5 epochs for *style* and the same number of epochs for *genre* labels. The network achieved a validation accuracy of **36.3**% when classifying on *style* and **66.0** % when classifying based on *genre*. For the Inception Score and top-1 accuracy, we use the *final* layer activations from the network, while for the F.I.D. we use the activations from the *third pooling layer* as suggested in [4].

In addition to quantitative evaluation, we also qualitatively compare our results to the outputs of the ArtGAN [11] paper to show differences in quality of generated images for each class.

4.4. Experimental Setup

The two stages of both the GANtor models were trained separately. The Stage-I was trained individually first, and was then connected to the rest of the network with its weights frozen during Stage-II training.

All experiments were run on 8 Nvidia Tesla-K80 GPUs in parallel, with operations parallelized along the batch dimension. The training configurations for the two models we used were as follows:

- **GANtor-v1:** Stage-I was trained over 60 epochs with a batch size of 512, requiring approximately 12 hours to train, while Stage-II was trained for 45 epochs with a batch size of 128, requiring about 36 hours to train on the same setup.

- **GANtor-v2:** Stage-I was trained over 90 epochs with a batch size of 512, requiring about 8 hours of training. Stage-II was trained for a total of 80 epochs, or about 60 hours, with the weights preserved for intermediary epochs for comparison.

5. Data

We used the wikiart dataset available for download on github: https://github.com/cs-chan/ArtGAN/tree/master/WikiArt%20Dataset. The dataset contains roughly 80000 pieces of visual art extracted from wikiart.org categorized by artist, genre, and style and organized into train and validation sets. Wikiart is the largest public available dataset that contains around 80,000 annotated artwork in terms of genre, artist and style class.

However, not all the artwork are annotated in the 3 respective classes. To be specific, all artwork are annotated for the style class (27 styles total), but only 60,000 artwork annotated for the genre class, and only around 20,000 artwork are annotated for the artist class. The dataset has significant class imbalance for *genre* labels, as shown in figure 5.

Figure 14 in the Appendix shows a batch of 64 samples measuring $(256px)^2$from the training set, centered after cropping. The real images fed into the **Stage-I** GAN were cropped to $(64px)^2$, while those fed into the **Stage-II** GAN were $(256px)^2$ in size. Prior to cropping to the desired sizes for the network, the images were centered to ensure the final image fed in preserved adequate information regarding style. To reduce likelihood of overfitting, random horizontal flips with probability 0.5 were applied.

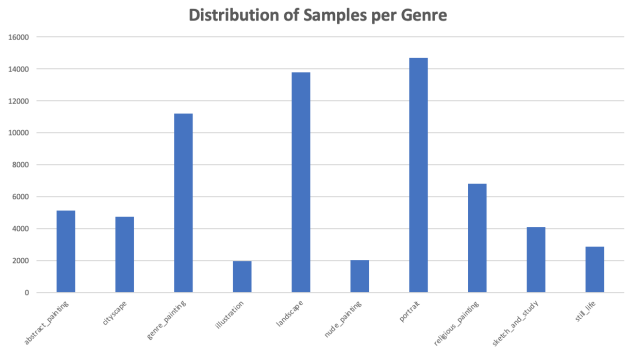

Figure 5: Distribution of the number of examples found in each genre. Note the extreme class imbalance.

6. Results and Discussion

6.1. Quantitative Evaluation

6.1.1 Inception Score and Accuracy

The results (table 1) show that the GANtor-v2 outperforms the baseline by 0.09 points in Inception Score. The peak classification accuracy of the GANtor-v2 model is also greater than the peak accuracy of GANtor-v1, indicating the more recent architecture has greater capacity to produce more distinguishable images.

Our results also show the most *distinguishable* results (i.e. highest inception accuracies) don't necessarily yield the highest inception scores. For Stage-II *genre* labels, the v-2 Generator yielded the highest I.S. after 60 epochs of training, by the Inception accuracy underwent a drop by over 8%. This indicates the samples within one class become *more* to distinguish from the next as training progresses beyond the 50th epoch, but the *saliency* of samples improves with training. This thesis is supported by the I.S. and top-1 accuracy comparison between epochs 50 and 80 in figure 6, which shows the classification accuracy peaks

Table 1: Inception Score (I.S.) and **Inception Score (I.S.) Top-1 accuracy** for each model and class category, generated using an **Inception-v3** trained on **Wikiart**

GANtor Model	Class category	Inception-v3 Metrics	
		I.S.	Top-1 (%)
v-1, Stage-I	genre	1.34 ± 0.08	21.47
v-2, Stage-I	genre	1.30 ± 0.04	10.16
v-1, Stage-II	style	1.35 ± 0.01	3.67
v-2, Stage-II	genre (epoch 50)	1.41 (epoch 50) $\pm$ 0.03	**17.46**
v-2, Stage-II	genre (epoch 60)	**1.44** $\pm$ **0.03**	9.05

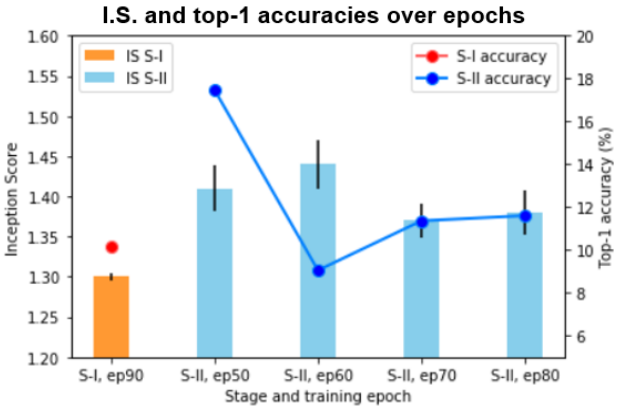

Figure 6: Comparison of Inception Scores and Inception-v3 top-1 accuracies for the last training epoch of the GANtor v-1 Stage-I generator and the last half of training for Stage-II. The class labels were based on *genre*.

where the I.S. drops, indicating that even though the overall diversity of artwork peaks at the 60th epoch *conditional*, the entropy of the *conditional* distribution goes down. This means that training beyond the 50th epoch decreases likelihood of getting art that belongs to the desired genre passed in through the conditioning vector, but increases the overall diversity in the generated art population.

6.1.2 Frechet Inception Distance

The F.I.D. metrics computed for the Stage-II generator results at epochs 50, 60, 70 and 80 (figure 7 and table 2) show a general increase in F.I.D. across training between the 50th and 80th epochs. This bolsters the general pattern of drop in I.S. from epoch 60 onward (figure 6). These two plots show the generator samples *worsen* in performance from epoch 60 onwards, which could have been due to instability

Table 2: F.I.D. for epochs 50 and 80 on the GANtor-v2 Stage-II outputs

Class category	F.I.D	
	Epoch 50	Epoch 80
abstract painting	136.75	148.4
cityscape	112.16	122.21
genre painting	92.75	93.96
illustration	122.12	131.08
landscape	90.28	95.77
nude painting	116.49	126.76
portrait	104.73	103.84
religious painting	102.65	107.51
sketch and study	113.7	123.8
still life	124.41	128.73

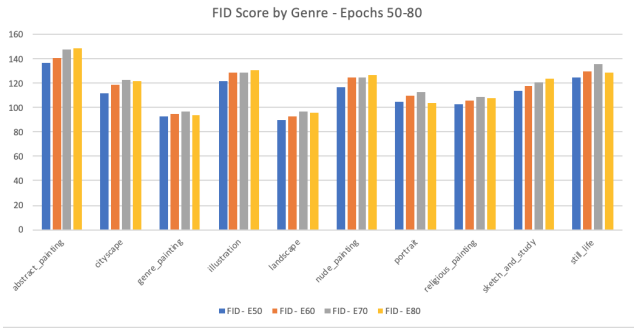

Figure 7: FID Score by genre at epochs 50 (blue), 60 (orange), 70 (grey), and 80 (yellow)

in training starting from somewhere between the 50th and 60th epochs.

6.1.3 Effect of Class Imbalance on F.I.D.

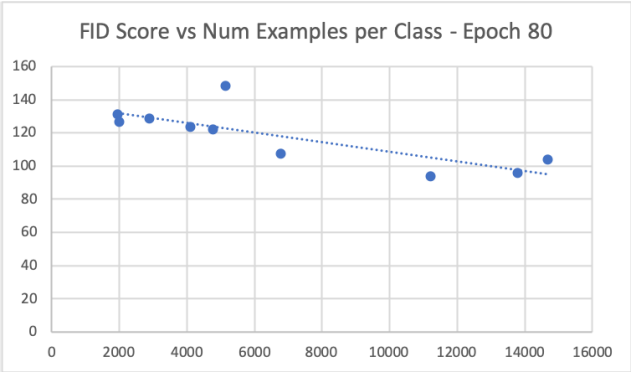

Figure 8: Negative correlation between the F.I.D. score and the number of examples available for each genre

One of the obstacles we faced with the Wikiart dataset was that of class imbalance between the genres. Figure 8 clearly shows a negative correlation between the F.I.D. score and number of examples present for each genre. This indicates that increasing the number of examples for some of the more sparse genres (example - *portraits*) could lead to the Stage-II generator producing more distinguishable samples for these classes, and increase the saliency of these samples in comparison to some of the classes the Generator performs well on (eg.- *landscapes*), both qualitatively (figure 4) and in terms of F.I.D.(figures 7 and 8). This could potentially mitigate the training instability for these sparse classes with training.

6.2. Discriminator performance during training

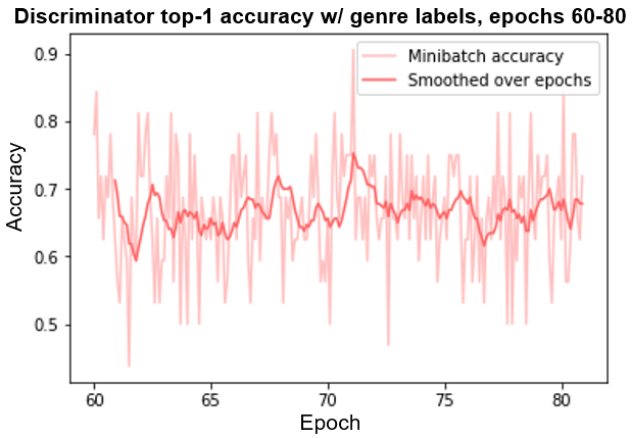

Figure 9: Accuracy for the v2 Stage-II Discriminator over training epochs

From epoch 60 onwards, our Stage-II discriminator reached average performance rivaling that used in ArtGAN. Their paper reported Discriminator accuracies around 66% at the end of training - our discriminator performed around that level (figure 9) from epoch 60 onwards, without showing a marked upward or downward trend, as would be expected from a well-balanced adversarial network. This shows that the instability in training highlighted in the previous sections was not a result of the Discriminator dominating the Generator.

6.3. Qualitative Analysis

6.3.1 Classification on style with GANtor-v1

Our baseline model was able to successfully output images clearly recognizable as artwork, and with a high level of variation between different outputs. Figure 15 gives a sample of 64 outputs from the Stage II Generator after the final epoch of training.

However, as one can see in Figure 10, while there is a high level of variation between outputs of our network,

from casual observation, one can see that there is little obvious separation between outputs conditioned on one category versus another. We believe that this is partially a result of the data, since there is a very high level of variation within each style category making it difficult for the model to separate based on style. This also likely explains why the IS scores we saw on this model were so low.

Generally speaking, in moving from the baseline to our v2 model, we wanted to improve differentiability between outputs while still maintaining the same quality of outputs, hence the incorporation of the ArtGAN discriminator and loss function with the StackGAN generators.

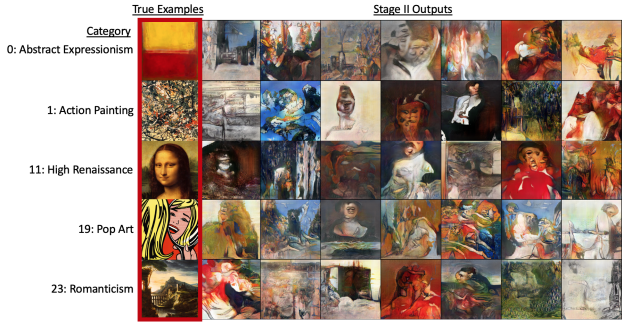

Figure 10: Stage-II Outputs by Style Category, Baseline model

6.3.2 Classification on genre with GANtor-v2

Similar to GANtor-v1, we saw that most images outputted by our model featured colors, patterns, and textures clearly recognizable as art, but with varying qualities of output. However, unlike v1, from a qualitative perspective, the works generated for each class were definitively more distinguishable, with, for example, many images generated for the *portrait* class showing humanoid forms,and the model performing especially admirably on landscapes. A sampling of generated images compared to real samples for each genre can be found in Figure 4.

Visual analysis of outputs generally supports the trends we saw in our IS and FID metrics. The genres that saw lower FID scores - *landscapes*, *genre painting*, and *portraits*, also saw in some very high quality outputs. This we suspect is a result of both the large number of sample images in these categories, along with the consistency across different examples in these classes and the relative semantic ease of generating these images. Meanwhile, due to the general falling trend in IS and rising trend in FID score from epochs 50 to 80 we suspect is because the generator by epoch 50 had learned how to output broader patterns that made the images easy to classify via an inception network, while the increasing detail of epoch 80 outputs hurt the network performance on quantitative metrics. However, like we mentioned earlier, we suspect that this was a result of increasing training instability after epoch 50.

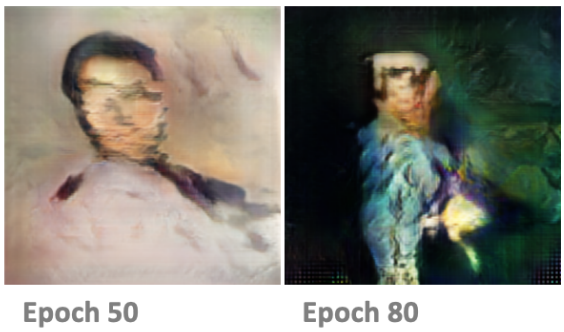

Figure 11: Stage-II portraits epoch 50 vs 80

7. Conclusion and Future Work

In this project, we successfully developed two Stacked Conditional Generative Adversarial Networks to generate artwork, GANtor-v1 and -v2. Using the labeled Wikiart dataset, we combined ideas from the **ArtGAN** and **StackGAN** projects to condition our generated art to different desired genres (for GANtor-v2) or styles(GANtor-v1) to varying degrees of success. Our final **GANtor-v2** architecture is a two-stage generative model that accepts a noise vector concatenated with a one-hot class vector and outputs a 256x256 image of the desired class. We evaluated our model outputs via the **Inception Score** and **Frechet Inception Distance**, and found that artwork generation is a very difficult task to quantitatively evaluate. However, using these metrics, we were able to deduce that our network performs much better on some classes (eg. *landscapes* and *genre painting*) compared to others (eg. *illustration* and *nude painting*), which was likely caused by the class imbalance in the dataset. We also showed that our model performance peaked between epochs 50 and 60 before encountering instability that damaged performance afterward. Thus, our best-performing model to generate art conditioned on *genre* labels using the existing Wikiart dataset would be the **GANtor-v2** with the Stage-II trained for **50 epochs**.

In the future, given more time and compute resources, we would like to experiment with different methods for training GANs - for example, implementing a **Wasserstein GAN**[1] that could stabilize the stage-II Generator training after 50 epochs. In addition, we would like to perform data augmentation on the dataset, especially for the classes that performed poorly on the F.I.D. Finally, we would like to explore different evaluation metrics other than the I.S. and F.I.D. in order to gain a more comprehensive understanding of our model's performance.

8. Contributions and Acknowledgements

8.1. Team Member Contributions

The development of the generative model architectures were equally split between Avoy and Michael.

Avoy handled the fine-tuning of the Inception-v3 and computation of the Inception Score and classification accuracies.

Michael primarily focused on the calculation of the FID score for the v2 model.

For the paper, Avoy mostly focused on the architectural descriptions while Michael did wonders with the graphics. The technical details were roughly split between the two.

8.2. Acknowledgements

Most of our original baseline architecture was adapted from code forked from the GitHub repo released by the authors of StackGAN available here: https://github.com/hanzhanggit/StackGAN-Pytorch. While we significantly changed almost every aspect of this code, the base structure remains the same.

Our code for the ArtGAN discriminator and loss was adapted from TensorFlow code released by the ArtGAN group here: https://github.com/cs-chan/ArtGAN/tree/master/ArtGAN

The code used to fine-tune the Inception-v3 network was based on https://github.com/pytorch/tutorials/blob/master/beginner_source/finetuning_torchvision_models_tutorial.py, and the function used to generate the Inception Score metric was obtained from https://github.com/sbarratt/inception-score-pytorch.

To generate our F.I.D. metrics, we adapted open-source available code available at: https://github.com/mseitzer/pytorch-fid

Finally, we would like to acknowledge Michael's CS229 project partner Jennie Yang for suggesting the name of this project. It wouldn't have been the same without her.

References

[1] Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein gan. 01 2017. 2, 8

[2] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018. 4

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 1

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 5

[5] Kenny Jones and Derrick Bonafilia. Gangogh: Creating art with gans, 2017. 2

[6] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951. 4

[7] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. 2

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. 1

[9] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. 4

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. 4, 5

[11] Wei Ren Tan, Chee Seng Chan, Hernán E. Aguirre, and Kiyoshi Tanaka. Artgan: Artwork synthesis with conditional categorial gans. *CoRR*, abs/1702.03410, 2017. 1, 2, 4, 5

[12] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5908–5916, Oct 2017. 2, 3

9. Appendix

Our github repository can be found at https://github.com/theCaiGuy/GANtor-Arts-Center

Generator	Classifier	Decoder
fcBN(512 × 4 × 4)	conv(128, 3, 2)	convBN(512, 3, 1)
NNupsample(8)	Dropout(0.2)	
convBN(512, 3, 1)	convBN(256, 3, 2)	NNupsample(8)
NNupsample(16)	Dropout(0.2)	convBN(256, 3, 1)
convBN(256, 3, 1)	convBN(512, 3, 2)	NNupsample(16)
NNupsample(32)	convBN(512, 3, 1)	convBN(128, 3, 1)
convBN(128, 3, 1)	Dropout(0.2)	
NNupsample(64)	convBN(1024, 3, 2)	NNupsample(32)
convBN(64, 3, 1)		convBN(64, 3, 1)
convBN(3, 3, 1)	fc(10)	NNupsample(64)
		convBN(32, 3, 1)
		conv(3, 3, 1)

Figure 12: Details for the GANtor-v2 architecture, with the Generator on the left and the Discriminator (decoder + classifier) on the right

Algorithm 1 Pseudocode for training ARTGAN

Require: Minibatch size, n and Learning rate, λ
Require: Randomly initialize θ_D and θ_G
Require: Denote parameters of Dec, $\theta_{Dec} \in \theta_g$
while condition not met **do**
  Sample $\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_n] \sim \mathcal{N}(0,1)^{n \times d}$
  Randomly set $\hat{\mathbf{Y}}_{\hat{k}} = [\hat{\mathbf{y}}_{\hat{k}_1}, \ldots, \hat{\mathbf{y}}_{\hat{k}_n}], \hat{k}_i \in \mathbf{K}$
  Sample minibatch $\mathbf{X}_r = [\mathbf{x}_r^1, \ldots, \mathbf{x}_r^n]$
  and $\mathbf{k} = [k_1, \ldots, k_n]$
  $\mathbf{Y} = D(\mathbf{X}_r)$
  $\hat{\mathbf{X}} = G(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}_{\hat{k}})$
  $\hat{\mathbf{Y}} = D(\hat{\mathbf{X}})$
  $\theta_D = \theta_D - \lambda \frac{\partial \mathcal{L}_D}{\partial \theta_D}, \mathcal{L}_D \leftarrow \mathbf{Y}, \mathbf{k}, \hat{\mathbf{Y}}, \hat{\mathbf{Y}}_{\hat{k}}$
  $\mathbf{Z} = Enc(\mathbf{X})$
  $\hat{\mathbf{X}}_z = Dec(\mathbf{Z})$
  $\theta_G = \theta_G - \lambda(\frac{\partial \mathcal{L}_{adv}}{\partial \theta_G} + \frac{\partial \mathcal{L}_{L2}}{\partial \theta_G}), \mathcal{L}_{adv} \leftarrow \hat{\mathbf{Y}}, \hat{\mathbf{Y}}_{\hat{k}}$,
    $\mathcal{L}_{L2} \leftarrow \mathbf{X}, \hat{\mathbf{X}}_z$
end while

Figure 13: Pseudocode for the algorithm used to train GANtor-v2

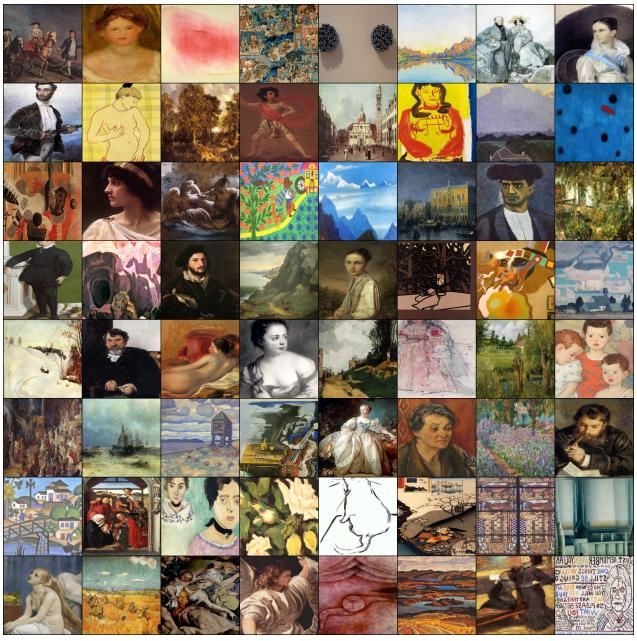

Figure 14: Samples from the **Wikiart** dataset, from a mix of 27 different styles

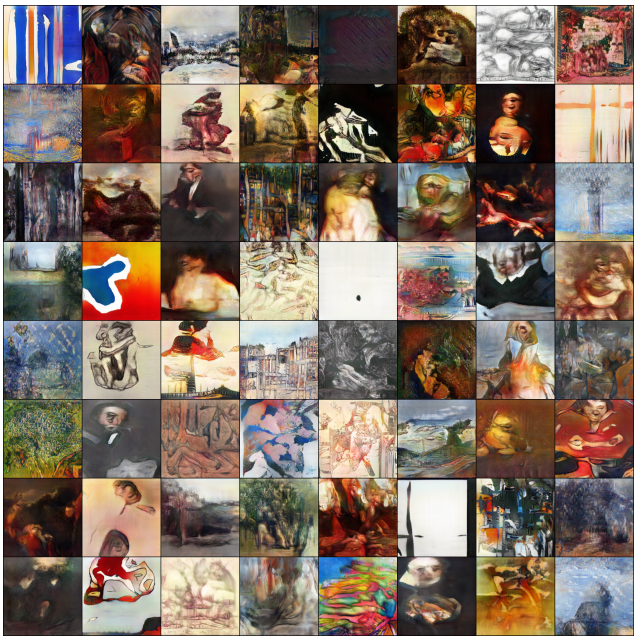

Figure 15: Baseline Stage II Outputs on Final Epoch.

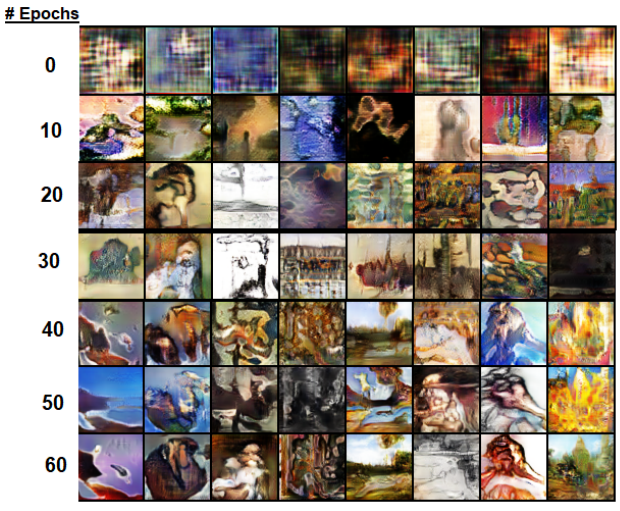

Figure 16: Progression of training output of Baseline Stage-I GAN

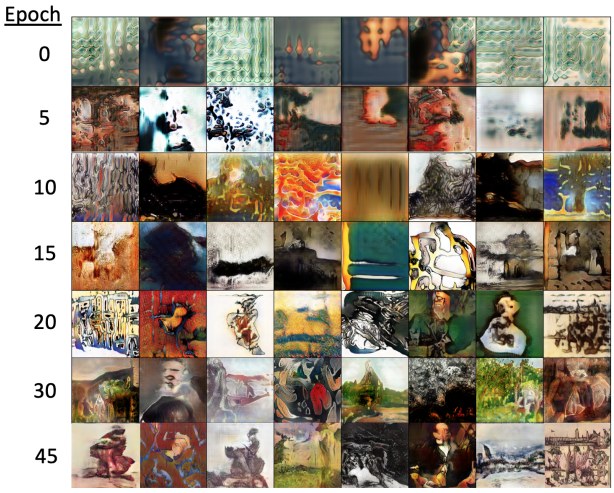

Figure 17: Progression of training output of Baseline Stage-II GAN

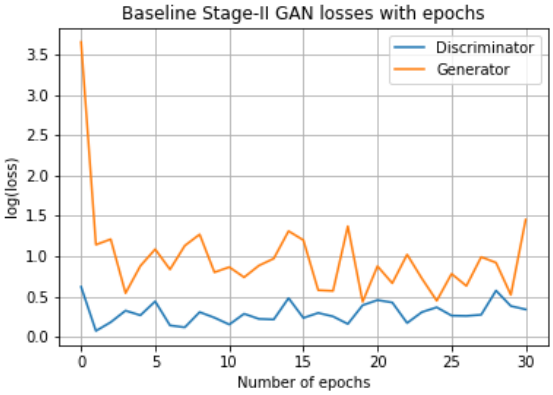

Figure 19: Stage-II GAN losses with number of training epochs

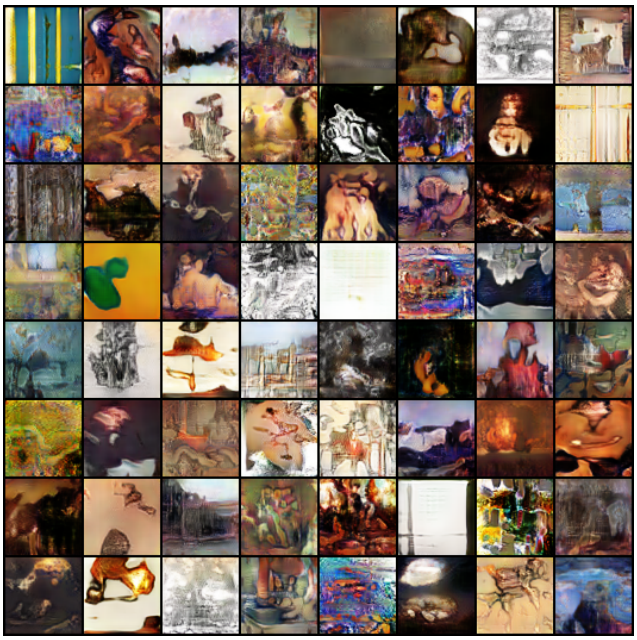

Figure 18: Baseline Stage II Inputs on Final Epoch (64x64 outputs from Stage I). Note how the Stage II GAN adds additional high-level features to the Stage I outputs

Figure 20: Solutions from the challenge presented at the beginning of our paper. Picures with a red heart were generated by our Stage-II generator. How well did you do?