Generation of Indian Arts using Generative Adversarial Networks (GANs)

Ashwin Milind Kulkarni*, Nimisha Pabbichetty†, Siddhi Pandare‡ and Aakash Venkataraman§
School of Electrical and Computer Engineering, Georgia Institute of Technology
Email: *akulkarni379@gatech.edu, †npabbichetty3@gatech.edu, ‡spandare3@gatech.edu, §avenkata44@gatech.edu

***Abstract*—The best way to show that artificial intelligence models created using machine learning and deep learning are intelligent is by testing their ability to create. Research-driven this thought in mind has led to the development of models that generate human-level creative products such as art. The beauty of human creativity lies in its intrinsic uniqueness resulting from the individual creating it and the environment that helped foster it. Cultural and geographic differences have led to the creation of a plethora of different art forms, and it was clear to us that the AI models should be trained to create unique and culturally specific art. This work is an effort in this direction as we investigate the capabilities of several GANs for generating Indian Art.**

***Index Terms*—Generative Adverserial Networks, art generation, Machine learning**

I. Introduction

Unsupervised models that summarize distribution of input data can be used to generate new examples in the input distribution. These models are called "Generative Models." Generative Adversarial Networks (or GANs) are Generative Models that use Deep Learning methods for their training. A GAN model consists of two sub-models:

1) Generator Model: Generates new examples
2) Discriminator Model: Checks whether a generated example is real (belongs to the input distribution) or fake (generated by the Generator)

GAN training is based on a zero-sum game scenario where the loss of one model implies a gain of the other model. If the generator generates an example that is classified as fake by the discriminator, the discriminator wins, and the generator loses which results in the generator model's parameters being penalized. On the other hand, if the generator generates an example that the discriminator fails to classify as fake, then the generator wins, and the discriminator loses, which results in the discriminator model's parameters being penalized. This keeps happening till the Nash equilibrium is reached and the samples being produced by the generator is indistinguishable from the training samples [1]. The methodology section explains the training procedure of a GAN in detail.

A. Deep Convolutional GAN (DCGAN)

DCGAN explicitly uses Convolutional Neural Networks (CNNs) for its Discriminator and Generator models, which makes DCGAN useful for image generation because of the success of CNNs in areas such as Computer Vision that deal with image data. Specifically, DCGAN utilizes Convolutional and Transposed Convolutional layers in its architecture. Convolutional layers extract meaningful information from an image. A convolution uses a filter to extract information from an image. Transposed Convolutional Layers filter the data but are used to upsample the data, i.e., to a larger output feature vector. Convolutions downsample the data, i.e., reduce the features to extract only relevant information. Usually, in a DCGAN, the Discriminator consists of convolutions, whereas the Generator consists of Transposed Convolutions.

B. Least Squares GAN (LSGAN)

Least Squares Generative Adversarial Networks (LS-GANs) is a variant of the GAN architecture that uses least squares loss functions instead of the more commonly used binary cross-entropy loss functions. In a standard GAN, the loss function used by the discriminator network is the binary cross-entropy loss, which is optimized by minimizing the difference between the predicted and true labels of the samples. However, this loss function can be unstable and difficult to optimize which can make the GAN difficult to train. In contrast, the least squares loss function used by LS-GANs is a smooth, continuous function that is easy to optimize, which makes the GAN training more stable and efficient. This can lead to better-behaved models and more realistic generated outputs. [2] [3]

C. GANGogh

GANgogh [4] is based on the Auxiliary Classifier GAN (AC-GAN) and the Wasserstein GAN (WGAN). The AC-GAN [5] improves upon the fundamental GAN architecture by adding a classification layer to the discriminator and a conditioning vector to the generator. So in addition to training the discriminator to maximize the probability of identifying a generated sample from a training sample, it is also trained to minimize the classification error. As a result, the generator can generate images that belong to different classes at the same time (as opposed to other GANs where they need to be individually trained for each class). GANGogh is set up under the Wasserstein model [6] along with the classification component. Pre-training was used as the model tended to initially focus on differentiating between 'real' and 'fake' samples and only trying to differentiate between the different classes in later stages of training. This was likely due to the discriminator being unable to differentiate between the different classes in the initial stages of training as it hadn't learned that information yet. This prevents the generator from differentiating between classes as well. So by pre-training our

discriminator model, it has an initial idea of the difference between classes. Global conditioning is employed to help the generator better differentiate the images it is producing as a result of the conditioning vector [7], [8]. Every time the activation functions is used, the conditioning vector can be used to influence the way the activation is applied. The conditional gated multiplicative activation function is used.

D. CycleGAN

CycleGAN is an unsupervised training technique of image translation models using the GAN architecture with the help of unpaired collections of images from two different domains. It automatically trains the image-to-image translation models without paired data samples. The generator takes in real images and outputs fake images, and the discriminator classifies whether the image is real or fake. LSGAN loss function and Cycle consistency loss were mainly used to optimize the performance of the discriminator and generator. The inability of the generators to be able to first map from one class domain to the other class domain, and then back to the first class domain without any changes to the image is quantified using cycle consistency loss. The generator has three main components, the encoding phase, the transformation, and the decoding. The encoding phase converts the features in the image to a latent space representation using multiple convolution layers. The transformation phase is comprised of six Resnet blocks and is used to assemble the appropriate latent features captured in the encoding phase. The decoding phase assembles the latent representations using transpose convolutions. The discriminator is a standard convolution neural network, which performs the binary classification. It decides if the image is real or fake.

II. Methodology

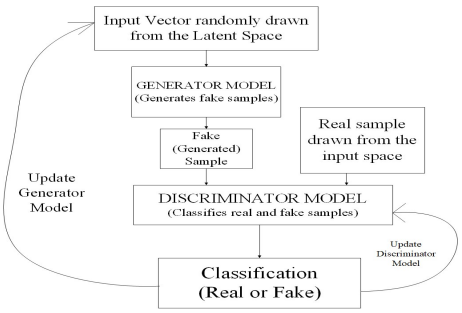


Fig. 1: Training Procedure of a GAN

Figure 1 shows an overview of the training procedure of a GAN. A vector is randomly drawn from the Latent Space of the input data. Latent Space consists of reduced feature vectors where similar data points are positioned close to each other [9]. The Generator model uses a vector from the latent space and generates fake images from it. These fake images, along with real images from the input dataset are fed to the Discriminator model which tries to classify real images as real and fake images as fake by trying to maximize the probability of assigning the correct label to both the generated and the training samples. Based on the number of correct classifications of the discriminator, both the models are penalized by heavily updating the weights of their respective Neural Networks [10]. In this way, the GAN is trained where Generator and Discriminator Models are trained as a minimax game.

Getting the dataset is the most important phase in the GAN training process. We created a dataset with 8 different art styles namely - Tanjore, Phad, Warli, Pattachitra, Gond, Kalighat, Mural and Madhubani. The images were scraped from different search engines like Bing, DuckGO and Flickr. Duplicates and outliers like pictures of the paintings hanging on walls etc were removed. To ensure that the dataset is balanced i.e all classes are equally represented and to add some variance to the dataset, a data augmentation pipeline was set up. The following techniques were applied - noise addition, horizontal flips, blurring and variations in hue and saturation. The final dataset statistics can be seen below in Table 1. All GANs are trained for 10000 epochs. [1]

TABLE I: Indian Art Generation Dataset

Class	Number of Images
Gond	628
Kalighat	636
Madhubani	616
Mural	669
Pattachitra	630
Phad	504
Tanjore	554
Warli	534
Total	**4771**

A. DCGAN Training

The Discriminator and Generator are trained together as shown in Figure 1.

1) Latent Space: The dimension of the latent space is chosen to be 100. The generator uses random vectors from this space to generate fake images.

2) Generator Model: The generator model consists of a Convolutional Neural Network with 4 hidden layers and 1 output layer. The tanh activation function is used for the generator as opposed to sigmoid since tanh has a greater gradient which results in a faster learning rate as compared to sigmoid activation.

3) Discriminator: The discriminator model also consists of a Convolutional Neural Network with 3 hidden layers and 1 output layer. Sigmoid activation is used for the discriminator with a binary cross entropy loss function and an Adam optimizer. After experimenting with various learning rates, we observed an optimal learning rate of 0.002.

[1]The link to the dataset: https://github.com/Nimisha-Pabbichetty/IndianArtGeneration.git

B. LSGAN Training

1) Latent Space: The dimension of the latent space is chosen to be 100.

2) Generator Model: The generator network of LSGAN consists of a series of transposed convolutional layers, which are used to upsample the input noise vectors to produce fake samples with the same dimensions as the real samples. In our experiments, we have used three convolutional layers that take $64 * 64$ input images. These layers are followed by batch normalization layers, which help to stabilize the training of the network by normalizing the activations of each layer. A ReLU activation function is then applied. At the output a Tanh activation function is used.

3) Discriminator Model: The discriminator network also consists of a series of convolutional layers, which are used to extract features from the input samples and determine whether they are real or fake. These convolutional layers are followed by batch normalization layers and activation functions, in a similar manner to the generator network. The only difference is the model is compiled with an $L2$ loss.

4) Experimentation Details: The visual quality of the images generated by GANs depends on the tuning of various parameters. We experimented with parameters such as learning rate, using different optimizers, and also tuning the βs for optimizers such as ADAM. While keeping the loss function constant that is $L2$, the experiments concluded that both ADAM (β_1 = 0.5 and β_2 = 0.99) and RMSProp give comparable results in image generation. For LSGANs, a learning rate of less than 0.0002 minimizes the loss to 0.3 for the generator and 0.15 for the discriminator at 10000 epochs. The batch size of 64 gives an acceptable tradeoff between the speed and stability of the learning process. [11]

C. GANGogh Training

1) Latent Space: The dimension of the latent space is chosen to be 128.

2) Generator Model: The generator model consists of a Neural Network with 6 convolutional layers that are alternated with a layer block that performs deconvolution, layer normalisation and the global conditioned activation function. The output is a 64*64 image.

3) Discriminator: The discriminator model consists of a Neural Network with 5 convolutional layers and one linear layer followed by the output layer. It takes in a 64*64 image and outputs the probability of it being 'real' or 'fake' and the class output.

4) Experimentation Details: Three different optimizers were experimented with - RMSProp, AdaGrad and Adam. It was observed that Adam performed the best while AdaGrad was the worst. This was likely due to AdaGrad's tendency to decay the learning rate aggressively which is solved in RMSProp and Adam. Three different learnring rates were also used - 0.001, 0.0001, 0.00001 and it was observed that 0.0001 gave better results. Three batch sizes - 8, 16 and 64 - was used, while training time was fastest when batch size was 8, it also meant that less information was being gained for each iteration requiring longer train time. It would be best to have even higher image resolutions, batch sizes and training durations to get the best possible results.

Class of Art	DCGAN	LSGAN	GanGogh	CycleGAN
Gond	0.3722	0.2257	0.1931	0.5081
Kalighat	0.2113	0.1154	0.2786	0.4812
Madhubani	0.242	0.3725	0.2047	0.3211
Mural	0.2332	0.1265	0.2056	0.2122
Pattachitra	0.2725	0.3376	0.3243	0.3721
Phad	0.2944	0.1559	0.2435	0.1219
Tanjore	0.4893	0.5463	0.4231	0.7842
Warli	0.3266	0.2771	0.2072	0.3241

TABLE II: SSIM for all classes of Arts for the chosen GANs

D. CycleGAN Training

1) Generator Model: The generator model consists of a class constructor where one layer is conv2D and one convTranspose 2D layer and there is a ReLu activation function. The output is a 64*64 image, kernel size is 7, padding is 3, number of features is 64 and padding mode is reflect. There is also down blocks in generator, kernel size is 7 and padding is 1.

2) Discriminator: The discriminator model consists of a class constructor where one layer is conv2D and one convTranspose 2D layer and there is a Leaky ReLu activation function. The output is a 64*64 image, kernel size is 4.. There is also down blocks in the discriminator, kernel size is 7 and padding is 1.

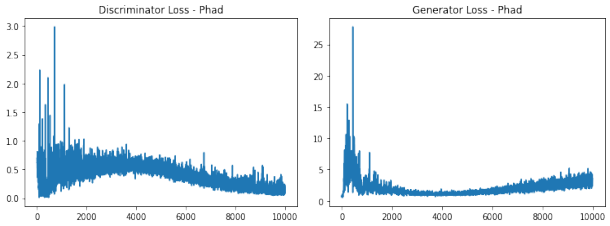
3) Training and Experimentation Details: The learning rate is varied from a wide set of values between 0.5 - 0.999 and $1e^{-5}$,$1e^{-7}$, $1e^{-9}$, and $1e^{-11}$. The model performed best with $1e^{-7}$. Testing of the model is also done for different batch sizes. As more images are trained in every batch, the time taken for the model to converge decreases. $lambda_A$ and $lambda_B$ which are weights for cycle-consistency loss (regularization term). A bigger lambda led to better reconstruction loss, but models made smaller changes in the generated images.

III. Results

The structural similarity index measure (SSIM) is a metric used to assess the similarity between two images. It takes into account the luminance, contrast, and structure of the images to calculate a value between -1 and 1, where 1 indicates a perfect match and -1 indicates a complete mismatch. The SSIM is commonly used to evaluate the images generated by the GANs against the training dataset. [12].The pictures generated by the different GANs are attached in the Appendix VI.

A. DCGAN

From Fig 2, we can observe that discriminator loss increases in the middle (around 3000-5000 epochs), and then decreases as it learns more. Since this is a zero-sum game, the generator loss follows an opposite trend. It decreases in the middle, and then slightly increases at the end of 10000 epochs. However, when trained for more epochs, the generator loss becomes less than the discriminator loss. The general shape of these graphs



a)Discriminator Loss b)Generator Loss

Fig. 2: Loss figures for Phad Arts by DCGAN

is constant across different Indian arts. However, the actual loss values and epoch number for minimum Generator loss varies. For e.g., from Figure 3, we can see that for Warli arts, the epoch number for minimum Generator loss is around 6000 while for Phad arts, it is around 4000.

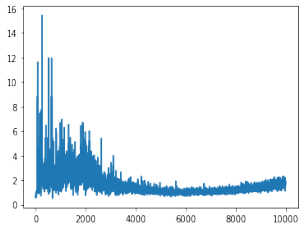
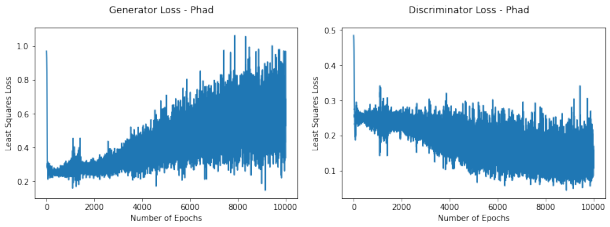


Fig. 3: Generator Loss for Warli arts

B. LSGAN

For all the classes of Indian arts (losses for Phad art are shown in Fig 4), the Generator loss converges around 0.5 while the discriminator loss converges to 0.2 for 10000 epochs. Although we see the art taking shape at this epoch, the discriminator can still predict well that it is fake. Even though LSGAN shows promising results in generating less complex images like MNIST numbers dataset, and road signs [13], LSGAN fails to generate satisfactory fake images for complex art forms like Indian Art.

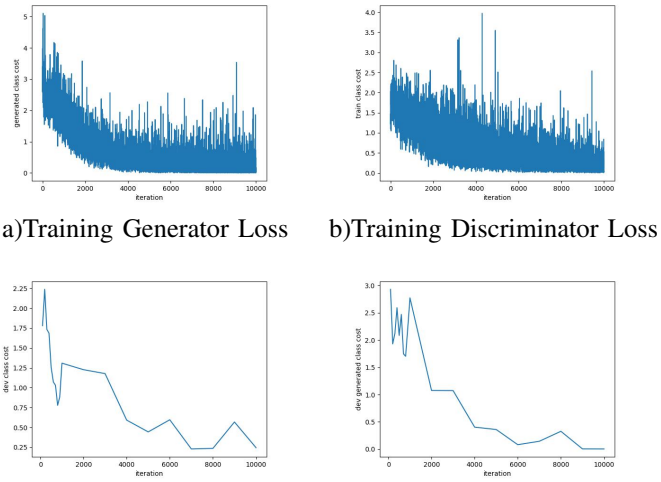


a) Generator Loss b) Discriminator Loss

Fig. 4: Loss figures for Phad Arts by LSGAN

C. GANGogh

In the training process of the GANGogh model, a validation step also exists. It occurs every 100 iterations for the first 1000 iterations and then every 1000 iterations from that point on. 1 in 20 samples (5%) of training data is used for validation. The



a)Training Generator Loss b)Training Discriminator Loss

c)Validation Discriminator Loss d)Validation Generator Loss

Fig. 5: Loss figures

losses of the model during training and validation can be seen in Fig 5. The settings for this run: epochs - 10000, learning rate - 0.0001, batch size - 64, optimizer - Adam.

D. CycleGAN

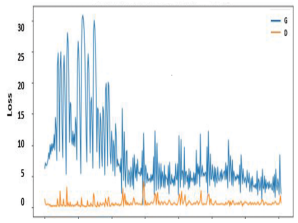


Fig. 6: Generator and Discriminator Loss figure for Cycle GAN (Tanjore art)

In the training process of the CycleGAN model, results are stored every 200 epochs. 1 in 5 samples of both training sets is used for the validation. As shown in the above image, the loss changes erratically for the initial epochs but later it converges when the Hfake (Cycle consistency loss) reaches the mark of around 0.533.

IV. Conclusion

We curated a dataset containing Indian art from 8 different classes and tested 4 types of GANs on it. We observed that the paintings were beginning to take shape when observed at the 10,000th epoch. With access to increased computational power, the GANs are able to generate similar paintings. Out of the 8 classes considered, all the GANs were best at generating Tanjore art. This could be attributed to the distinct structure of the art itself, allowing it to be captured easily by the models. This was proved quantitatively as the SSIM metric was the largest for Tanjore art. Given the constraints, CycleGAN performs the best as it does not learn the features of the training image set from scratch.

REFERENCES

[1] Goodfellow, Ian J. and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua "Generative Adversarial Networks" arXiv, 2014. https://arxiv.org/abs/1406.2661

[2] Mao, Xudong, et al. "Least Squares Generative Adversarial Networks." arXiv preprint arXiv:1611.04076, 2017.

[3] Pieters, M., Wiering, M.A. (2018). Comparing Generative Adversarial Network Techniques for Image Creation and Modification. ArXiv, abs/1803.09093.

[4] Kenny Jones and Derrick Bonafilia, GANGogh: Creating Art with GANs, https://towardsdatascience.com/gangogh-creating-art-with-gans-8d087d8f74a1

[5] Odena, Augustus and Olah, Christopher and Shlens, Jonathon "Conditional Image Synthesis With Auxiliary Classifier GANs"arXiv, 2014. https://doi.org/10.48550/arxiv.1610.09585.

[6] Arjovsky, Martin and Chintala, Soumith and Bottou, Léon, "Wasserstein GAN" arXiv, 2017. https://doi.org/10.48550/arxiv.1701.07875

[7] Oord, Aaron van den and Dieleman, Sander and Zen, Heiga and Simonyan, Karen and Vinyals, Oriol and Graves, Alex and Kalchbrenner, Nal and Senior, Andrew and Kavukcuoglu, Koray "WaveNet: A Generative Model for Raw Audio", arXiv, 2016. https://doi.org/10.48550/arxiv.1609.03499

[8] Oord, Aaron van den and Kalchbrenner, Nal and Vinyals, Oriol and Espeholt, Lasse and Graves, Alex and Kavukcuoglu, Koray, "Conditional Image Generation with PixelCNN Decoders" arXiv, 2016. https://doi.org/10.48550/arxiv.1606.05328

[9] https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/

[10] DCGAN - https://towardsdatascience.com/deep-convolutional-gan-how-to-use-a-dcgan-to-generate-images-in-python-b08afd4d124e

[11] Zhi, J. (2017). Pixelbrush: Art generation from text with gans. In Cl. Proj. Stanford CS231N Convolutional Neural Networks Vis. Recognition, Sprint 2017 (p. 256).

[12] Hassan, M., Bhagvati, C. (2012). Structural similarity measure for color images. International Journal of Computer Applications, 43(14), 7-12.

[13] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks by Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros

[14] Building a GAN for images - https://towardsdatascience.com/building-a-gan-with-pytorch-237b4b07ca9a

[15] Overview of Generative Adversarial Networks - https://realpython.com/generative-adversarial-networks/

[16] A DCGAN Tutorial - https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

[17] A Gentle Introduction to Generative Adversarial Networks (GANs) - https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

[18] Dewi, C.; Chen, R.-C.; Liu, Y.-T.; Yu, H. Various Generative Adversarial Networks Model for Synthetic Prohibitory Sign Image Generation. Appl. Sci. 2021, 11, 2913. https://doi.org/10.3390/app11072913

[19] Application of Improved CycleGAN in Laser-Visible Face Image Translation. Mingyu Qin,1 Youchen Fan,2,* Huichao Guo,3 and Mingqian Wang1

[20] Embedded CycleGAN for Shape-Agnostic Image-to-Image Translation. Ram Longman

V. Per Member Contribution

Every member was responsible for performing in-depth research, training respective GANs and making conclusions from the results. Members responsible for the different GANs are as follows:-

- **DCGAN - Ashwin Kulkarni**
- **LSGAN - Siddhi Pandare**
- **GanGogh - Nimisha Pabichetty**
- **CycleGAN - Aakash Venkatraman**

VI. Appendix

A. *Original images*

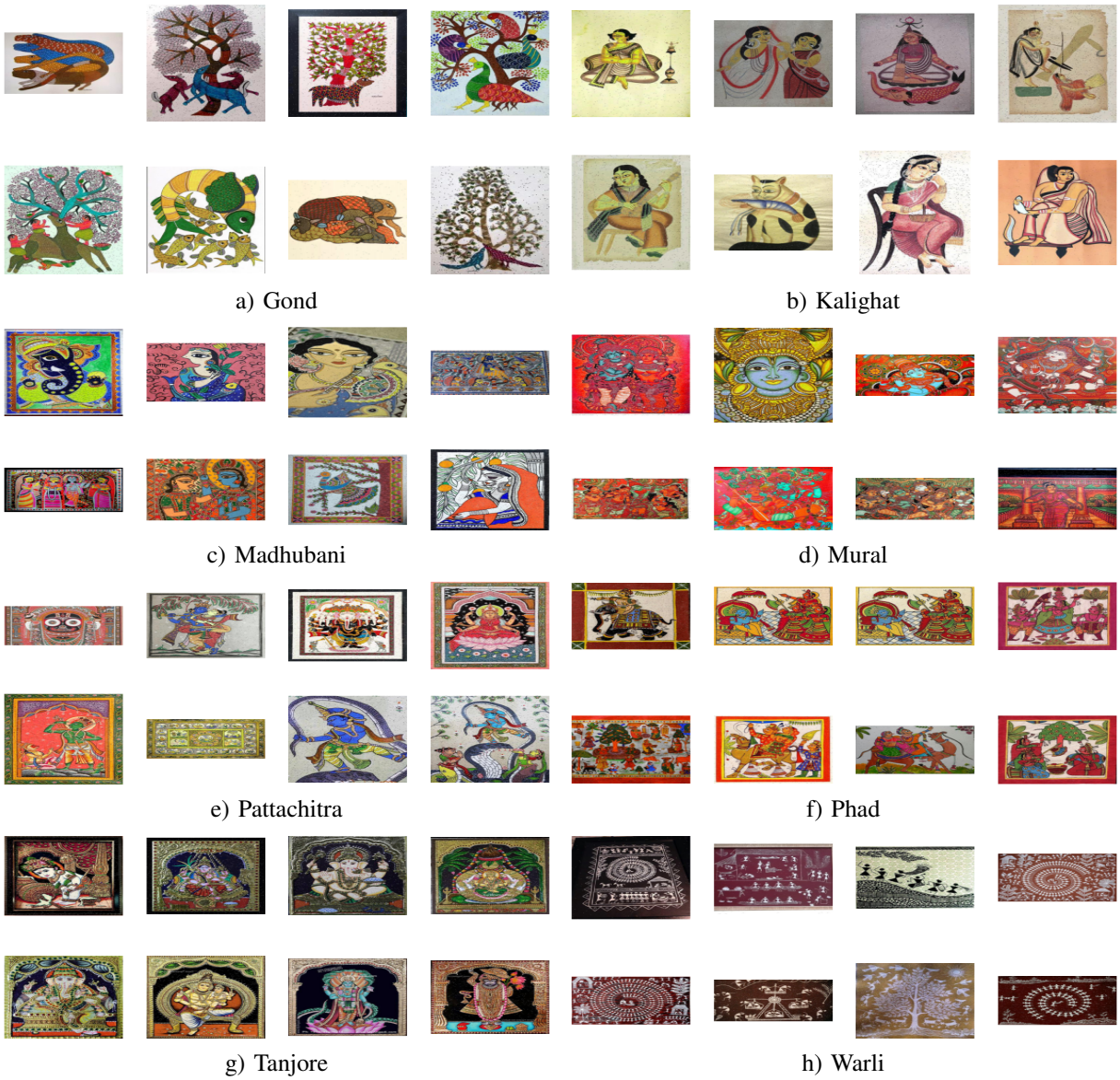
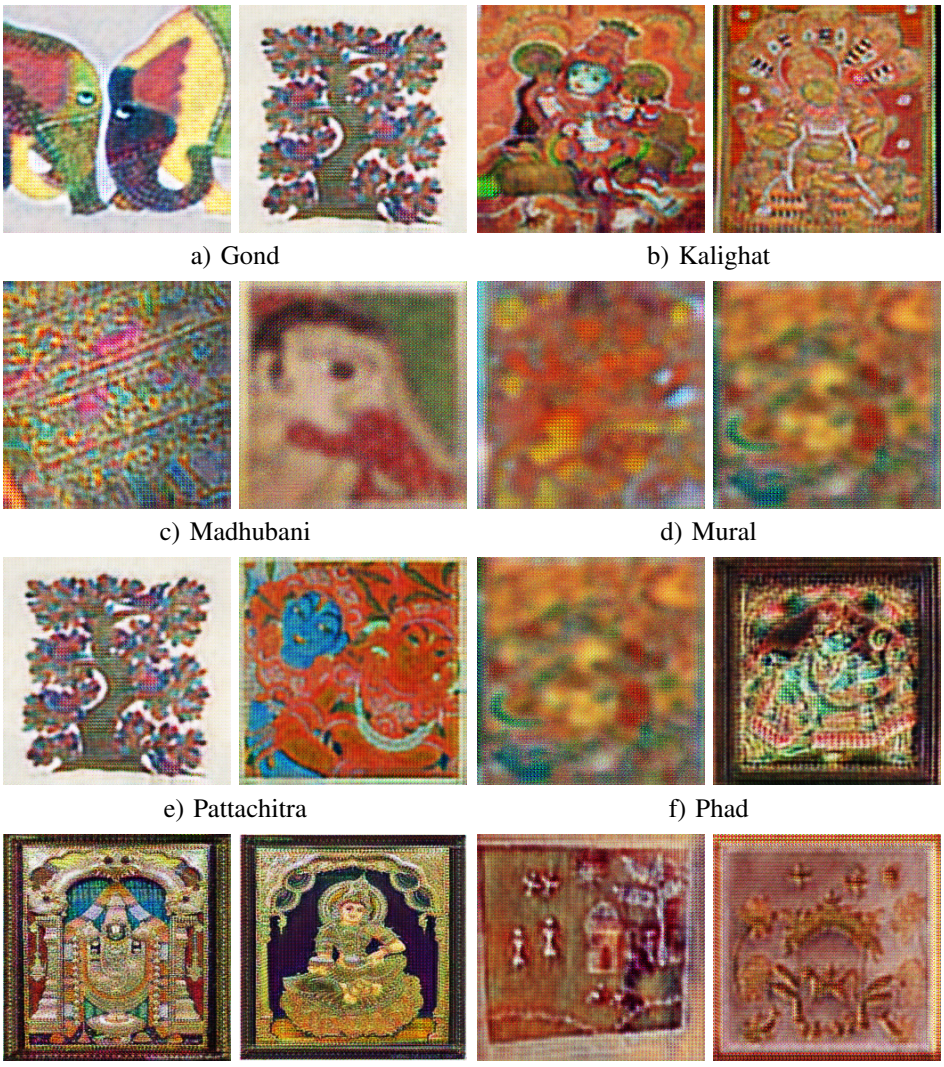


Fig. 7: Original images of the 8 classes of Indian Art used in the training process.

B. CycleGAN Results



a) Gond b) Kalighat

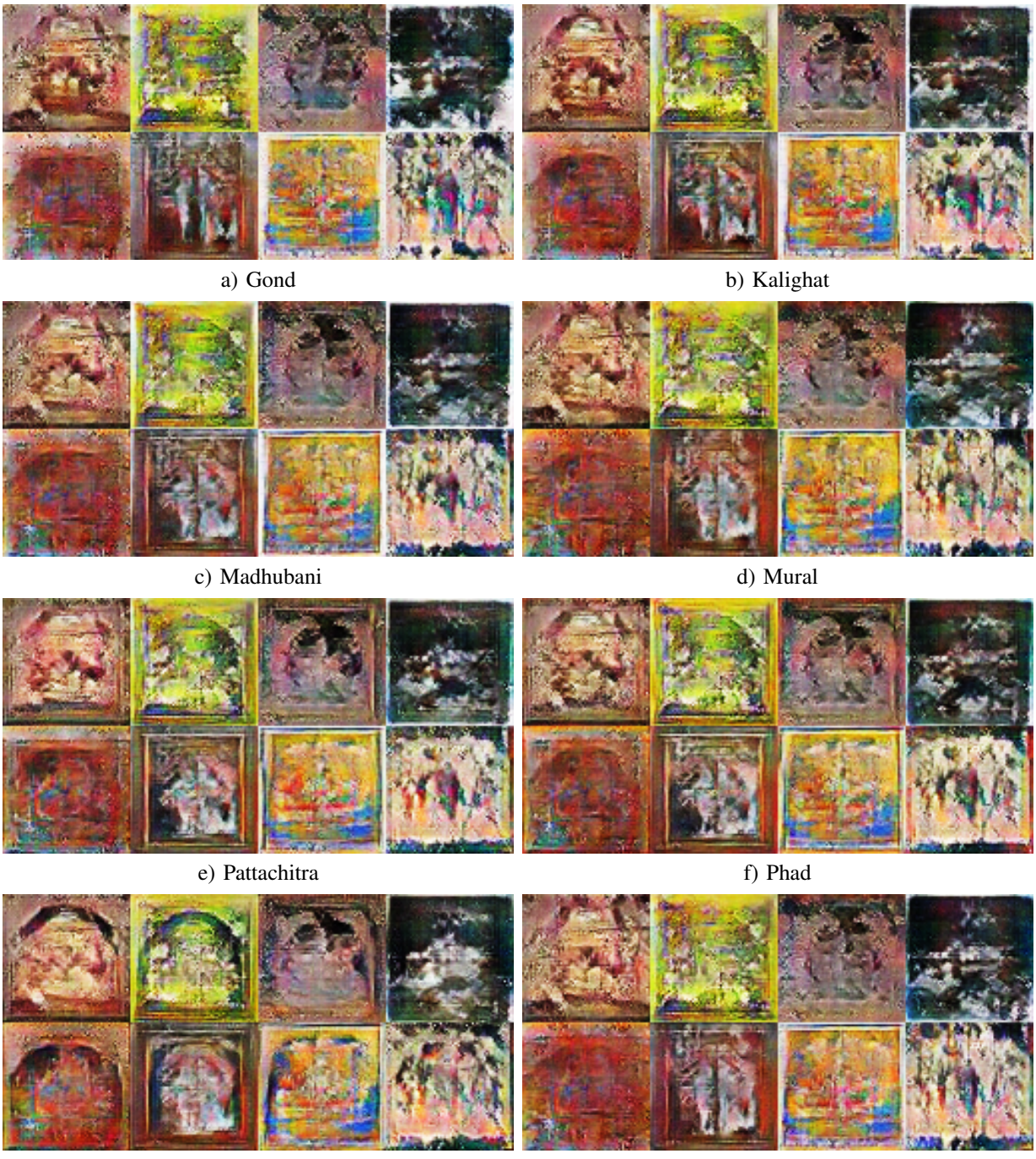
c) Madhubani d) Mural

e) Pattachitra f) Phad

g) Tanjore h) Warli

Fig. 8: Outputs generated by the CycleGAN model. The paintings are beginning to take shape. Tanjore and Kalighat especially are better defined because of the prevalence of borders in their style.

C. GanGogh Results



a) Gond b) Kalighat

c) Madhubani d) Mural

e) Pattachitra f) Phad

g) Tanjore h) Warli

Fig. 9: Outputs generated by the GANGogh model. The paintings are beginning to take shape. Tanjore and Pattachitra especially are better defined because of the prevalence of borders in their style.

D. DCGAN Results

Fig. 10: Outputs generated by the DCGAN model. Paintings belonging to Gond and Warli categories give most accurate results as compared to others. If the GAN is trained for around 100,000 epochs, the accuracy would be huge

E. LSGAN Results

a) Gond b)Kalighat

c) Tanjore d) Mural

e) Pattachitra f) Phad

g) Warli h) Madhubhani

Fig. 11: Outputs generated by the LSGAN model. The paintings are beginning to take shape. Tanjore, Gond, and Kalighat especially are better defined because of the prevalence of borders in their style.