Ast-GAN: Art Style Generation and Transfer

G030 (Mohamad Harah, Yazan Ali, Feysal Kethuda)

Abstract

In this paper, we explore the task of learning the style of a group of paintings and applying on other images to make them have the same style. We used the Wiki-art Landscapes dataset. We achieved this by combining architecture of Generative Adversely Network (GAN) with a Neural Style Transfer stage for generating styles and applying them to real world images to produce novel artworks. We found that weakening the discriminator helps generate better output for our use-case. Our experiments provided a very unique insight of how styles could be applied to images through the transfer of colours and designs while maintaining the original outline and shape of the content image.

1. Introduction

With the breakthroughs happening in the field of Generative Adversarial Networks (GANs), it is now possible to have networks create art from scratch. Other networks have proved capable of transferring the style of one painting to another. However, combining the two techniques to learn and create similar styles to apply to our pictures is something that has not been done. And do we not all wonder how the pictures we take would look if they were drawn by Picasso or Monet?

The main motivation behind this project is that we believe we could combine the advantages of several well researched techniques in a single algorithm to produce more artworks of a specific style such as a painter's style or a genre. We used popular GANS such as Deep Convolutional GAN (DC-GAN) in combination with Neural Style Transfer algorithm to learn image styles and apply them on other images.

With our technique we could learn the style of not just a painting but maybe the artists themselves such as Picasso and use that data to generate new paintings or apply the style to real images. We attempt to learn the style of the landscape images generated by the GAN and use neural style transfer to apply the style of this image to landscape pictures from the internet to produce novel artworks. We tested GANs and neural style transfer separately to check their basic functionality as well as the feasibility of our project.

By combining several techniques together, we eliminate some of their weaknesses and limitations and try to produce a novel technique of art generation which has nor been described or implemented before. In most of the papers these techniques have been implemented separately or on their own and therefore did not tackle the task of both learning to create similar styles and apply it to regular images or generate new ones from scratch.

2. Data set and task

For the GAN data we plan to use a collection of artworks that is publicly available called wiki arts . Moreover , for the neural style transfer we plan to use a collection of real pictures from the internet. As we plan to use data set of 15000 landscape images taken from wikiarts, we believe our implementation is not limited to random paintings , character classification , etc. Furthermore, to make the images compatible with the GAN for training , the images are pre-processed so that they 64 by 64 pixels and crop centred. Moreover, in our project we not only work on implementing this base model but we also try to improve the base model by trying more advanced algorithms and testing different hyperparameters which try to fix the shortcomings like vanishing gradient of a regular GAN.

The performance of the neural style transfer is validated by human analysis of the final output. This is because final quality of the product is subjective and depends on the opinion of the person viewing the image. However, there are some specific details and features we expected from the final output. For instance , in most of the examples presented in this paper we are able to see how the hues and patterns from the style image generated by the DCGAN have been applied on the content image while maintaining the content image's outline and original shape structures. This has allowed us to produce visually appealing pieces of unique artwork.

Moreover, the performance of the GAN is also validated with a similar method. In particular, our goal over here was to minimize the loss functions so that the generator is able to successfully trick the discriminator with generated artworks. In addition, similar to the neural style transfer, we visually analysed the final output to determine its quality and make any necessary adjustments or modifications in our system.

3. Methodology

Moreover, in our project we not only work on implementing this base model but we also try to improve the base model

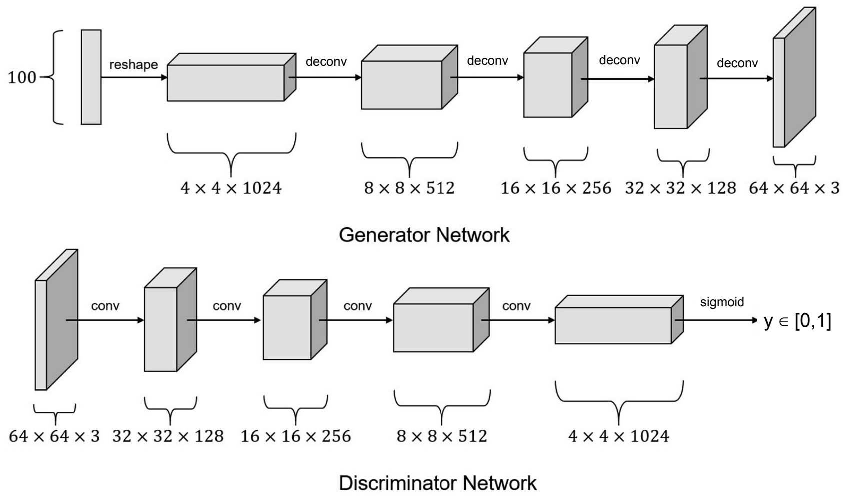

Figure 1. DCGAN Generator and Discriminator Convolutional Networks

by trying more advanced algorithm based on well known research papers.

3.1. DCGAN

We train a Deep Convolutional Generative Adversely Network (DCGAN) with the same architecture as (Radford et al., 2015). DCGANs are very similar to the original GANs described in (Goodfellow et al., 2014) but instead of relying on dense layers in each of the discriminator and the generator, we construct a network of strided 2 dimensional convolutional layers, 2 dimensional batch normalization layers and leaky ReLU activations for the discriminator, and for the generator we construct a network of 2 dimensional transposed convolutional layers, 2 dimensional batch normalization layers and ReLU activations and a tanh function at the end to normalize data to a range [-1, 1].

The generator's input is a latent vector that represents noise drawn from a standard normal distribution, and the output is a 3x64x64 image (3 is for RGB channels). The discriminator's input is a 3x64x64 image, and the output is a probability between 0 and 1 of whether the image is real or fake (1 being real and 0 being fake). The exact number of layers and layer sizes in both of the generator and discriminator are shown in figure 1. The main advantages of having such architecture over the original (Goodfellow et al., 2014) GANs architecture are that transposed convolutional layers in the generator can capture more details that can construct images from noise, and the convolutional layers in the discriminator are better at identifying discriminating features between fake and real images. Additionally, having batch normalization layers benefits the gradient flow through the network as prevents it from vanishing. (Ioffe & Szegedy, 2015).

DCGAN Loss function: We use the standard GAN loss function from (Goodfellow et al., 2014) given real image x, latent vector (noise) z, a discriminator D and a generator G.

$$L_{criterion} = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$$

The discriminator tries to maximize the above function and the generator tries to minimize it.

3.2. DCGAN Training

We train our DCGAN with 2e-4 learning rate, 0.5 beta1, 128 batch size, 0.3 negative slope for Leaky ReLU activations, input images with size 3x64x64 and the weights are initialized from normal distribution.

Part 1 - Training the discriminator:

We pass a batch of real images from training data forwards through the Discriminator D and calculate the loss $\log(D(x))$, calculate the gradients through backpropagation, pass a batch of fake images which are generated by the Generator G forward through the D, calculate the loss $\log(1 - D(G(z))$ and then we calculate the gradients through backpropagation. Lastly, we calculate the loss for the discriminator as sum over real batch loss and fake batch loss, then we update the discriminator's optimizer.

Part 2 - Training the generator:

maximize $\log(D(G(z)))$ by classifying the Generator output from Part 1 with the Discriminator, computing Generator's loss using real labels as ground truth, computing Generator's gradients with backpropagation, and finally updating Generator's parameters with an optimizer step.

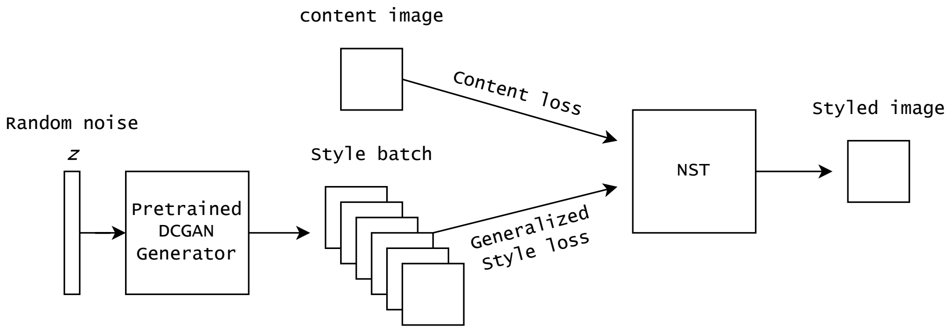

Figure 2. Transferring generalized style from generated batch on a content image

3.3. NST

After pretraining our DCGAN, we generate a batch of style images form DCGAN's generator and then pass it to our Neural Style Transfer Algorithm (NST) along with a content image to produce an image that has the content from the content image and a style that is generalized from the styles batch. The overall style generalization and transferring process is shown in figure 2.

Our Implementation of NST follows the original NST paper (Gatys et al., 2015) by utilizing a pretrained VGG19 on ImagNet. Moreover, we follow most of the original algorithm with slight adjustment to style loss calculation, where we average it over the entire batch to make a generalized style that can be transferred to another image.

The total loss for NST Given by:

$$L_{total} = \alpha L_{content} + \beta L_{style}$$

where α and β are loss weights for content and style respectively.
Content loss is the mean squared error or the squared distance between sets of n feature maps of an input image X and a content image C in a content layer, and we assign one layer from VGG19 model as the content layer, which is layer [conv_4]

$$L_{content} = \frac{1}{n}\sum_{i=1}^{n}(F_{XLi} - F_{CLi})^2$$

Style loss is the sum of the style loss in each style layer l of the model. In our case, the layers are added to [conv_1 conv_2, conv_3, conv_4, conv_5] of the VGG model layers.

$$L_{style} = \sum_{l=1}^{n} L_{style}^{l}$$

where style loss for each layer l for a batch of m style images is the average of style loss for all images in the batch:

$$L_{style}^{l} = \frac{1}{m}\sum_{i=1}^{m} L_{style_i}^{l}$$

And the style loss for a layer l for image i is the mean squared error between the gram matrix G of the style image S and input image X in layer l:

$$L_{style_i}^{l} = (G_i^l(S) - G_i^l(X))^2$$

4. Experiments

We have done several experiments to find the model's optimal configurations. Namely, we performed experiments to find the best generator base model, discriminator hyperparameter tuning, and style weight tuning for our Neural Style Transfer phase.

4.1. Generator Base Model

In this experiment, our aim was to find the most optimal baseline to use for our generator network. We considered using either a variational autoencoder GAN (VAEGAN) due to its architectural ability of projecting training images onto a latent space and sample that for new samples or a convolutional network as detailed in the DCGAN paper.

Since the purpose of this step is to choose a baseline model, we did not need a prior baseline. Instead, we compared the performance of the two models to each other after being trained on an equal number of epochs. We evaluated the model based on the two losses that were equivalent; the VAEGAN prior loss and Discriminator loss are similar to DCGAN's Generator loss and Discriminator loss respectively. We also visually analysed the output from both models to compare the images generated.

To carry out this experiment, we implemented both models as described in their original papers. We then changed the optimizer for the Variational auto-encoder GAN from RMSprop to Adam as the latter is widely considered better. Unsurprisingly, the Adam optimizer did prove superior in our case as well.

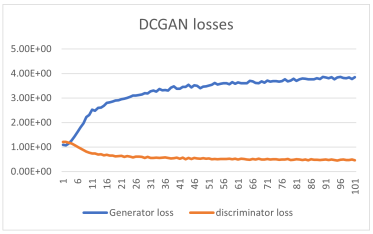

Figure 3. Plot of Generator and Discriminator losses of DC-GAN

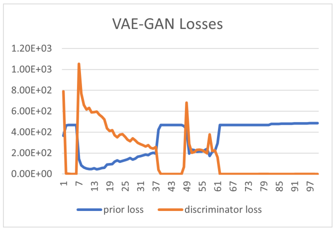

Figure 4. Plot of Prior and Discriminator losses of VAE-GAN

from looking at the loss plots in figures 3 and 4. We can see that in both these models the discriminator tends to get more powerful. We can also see that the DCGAN trains with much higher stability and converges at a lower loss.

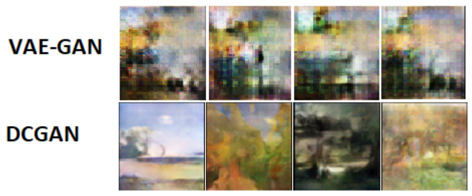

Figure 5. Output samples of DC-GAN and VAE-GAN after training for 100 epochs

Our results gave evidence of the fact that the DCGAN trains with more stability and converges faster than VAEGAN. Moreover, the images generated from DCGAN were mostly less blurry than the VAEGAN images. This is likely due to the generator having a simpler learning task which allowed loss to propagate more efficiently as can be seen from the images in figure 5.

4.2. Discriminator Hyperparameter Tuning

After choosing DCGAN as our baseline, we could observe that the Generator loss tends to increase while the discriminator just decreases. This is due to the discriminator being a lot more powerful and the generator has no idea of how to fool it anymore at that point. So, in this experiment, we aimed to find the balance between our generator and our discriminator networks. The imbalance between the two networks is a common problem with Generative Adversarial Networks.

We evaluated the effect of different hyperparameters by comparing the losses of the Generator and the Discriminator.

We experimented with changing the leaky coefficient for the Leaky RELU activations in the discriminator network. Increasing that value meant steeper gradients. After experimenting with the leaky coefficient, we found that 0.3 gave optimal results and coefficients of 0.4 or higher caused the discriminator to be too weak for our model.

We also performed label smoothing so that we can put an upper limit on the confidence of the discriminator to identify whether a sample is real. We did one sided smoothing and changed the real labels to be 0.85 instead of 1. This meant that the discriminator would never become too confident. We found that values less than 0.85 caused the discriminator to be too weak and that performance is improved if we reduce smoothing and change the label to 0.95 half-way through training.

Finally, we tried different optimizers and learning rates for the different networks. Specifically, we tried using RMSprop instead of Adam for the discriminator but that made the convergence too slow. So, we considered changing the learning rate of the discriminator however, increasing the learning rate caused learning to be very unstable. Furthermore, changing the learning rate for the discriminator to 0.1 * leaning rate of Generator ended up making the generator more powerful than the discriminator and in turn lead to low quality output.

After fine-tuning the hyperparameters, we compare the optimal network losses with the baseline. We then visually checked images generated by both of these models.

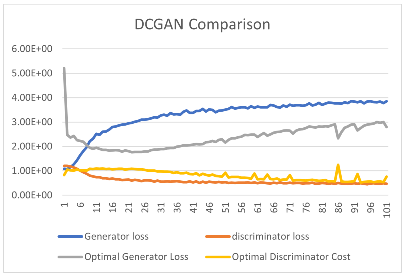

Figure 6. Plot of Generator and Discriminator losses of DC-GAN with default and weakened discriminator parameters

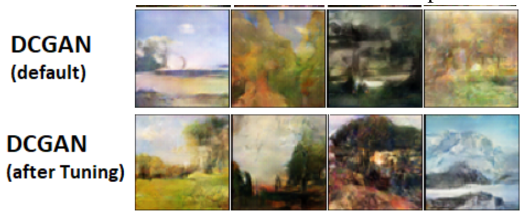

Figure 7. Output samples of DC-GAN with default and weakened discriminator parameters

the losses in plot 6 show that when we weaken the discriminator the discriminator loss increases slightly but the generator is smaller and takes longer to have the discriminator outperform it. We also noticed from the results in figure 7 that weakening the discriminator improves the quality of the generated images. The images generated from the default network are more blurry and have less defined landscape components than when the generator can learn from those signals

4.3. Style Weight Tuning

After training the model, we need to tune the hyperparameters of the style transfer component. There is one main hyperparameter and that is the style weight. This component specifies how much to focus on the style as opposed to keeping the original content.

The only way to evaluate this is through visual inspection and analysis.

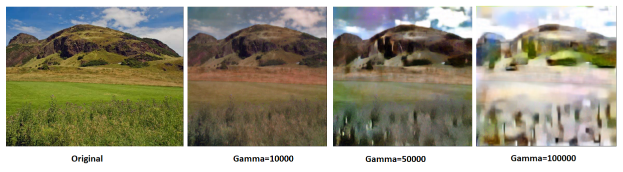

Figure 8. output of style transfer with various values of style weight (gamma)

when the style weight value was too high, it resulted in the styling being broken. And when it was too low then the style was not being transferred enough.

We found that 50,000 worked best on majority of the cases. Examples of various values can be seen in figure 8

5. Related work

Neural style transfer is a special type of optimization technique in which an image for content and another image for a style reference such as a painting by a famous artist are blended together to produce an output image which looks like the original content image, but is painted or created in the style of the reference image.

GANS stand for Generated Adversarial Networks. In a GANS two models are trained simultaneously – a generator which learns to produce images which look real and a discriminator which learns to discriminate real images from fake ones. So , as the model progresses, the generator and discriminator improve at their respective classifications until the discriminator is no longer able to differentiate between real and fake images. In DCGAN we make use of a deep convolutional network to train and test our model.

ST-VAE (Liu et al., 2021) explored using variational autoencoders in order to perform Neural Style Transfer. They mapped the content image and a small number of style images into a latent space which produced a quicker style transfer than in the original Gatys paper.

More recently, StyleGAN (Karras et al., 2018) proposed an alternative generator architecture for generative adversarial networks, borrowing from previous style transfer literature.

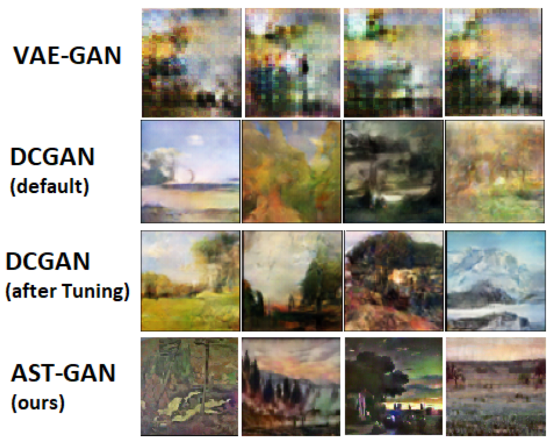

Figure 9. Comparison of artwork generated from our model at various stages

Their architecture leads to an automatically learned, unsupervised separation of high-level attributes. It differs from our model in that their model is much more computationally heavy and also is not used to transfer styles to new paintings.

Finally, CycleGAN (Zhu et al., 2017) is the state-of-the-art method for domain transfer between two groups of images. It does so by training two GANs that learn a two way mapping of images and is then able to translate images from one domain (style) to the other. CycleGAN however can not be used to generate new paintings of a certain style.

6. Conclusions

To conclude, our project allowed us to explore this unique idea of combining different state of the art techniques for producing unique pieces of art and answer our crucial research question about whether it's possible to learn artwork style with GANs and apply it to real world images. It's evident that the model we designed and implemented definitely has a lot of potential as from our results we can deduce that we are able to successfully generate and apply styles to real world images. If we try to analyse any of the several examples presented in this paper, we notice how colours and designs from the image generated by our trained DCGAN have been successfully painted upon the outline and the grid of the real-world content image through neural style transfer. This has given us the power to view the most popular images around us from a new perspective and in a unique style.

Moreover, even though this paper assisted us in implementing our own version of popular algorithms, there is still a lot of scope of further improvement and enhancements. One of the key fields where we believe we need to research further is the pixel quality of the final image produced. The current quality is not too high because the dataset used by our DCGAN for training contained images low quality 64 by 64 images. Some possible methods of overcoming this problem involves using more powerful Graphics Processing Units and training for longer periods of time (several days). Furthermore , we also explored the newly developed convolutional and variational autoencoders especially designed to enhance the resolution of images and improve pixel quality. However , due to lack of time and computational power we were not able to efficiently implement it. So , assuming access to more powerful machines and more time available to train this technique could also be integrated in our current model to generate more high quality and detailed artworks. The paper (Kumar & Goel, 2020) refers to and describes the autoencoder we mention as a possible extension.

Furthermore, another possible idea to explore could be using more technologically advanced GANS for the training process. This is because in the training part we are attempting to learn the style of the artist or the group of artworks. However , some GANS have the issue of gradient loss , model collapse , lack of stability , etc. which affects the quality of the final product. Through research one possible fix we recommend is using the W-GANS which are known to fix this issue by modifying the distance and loss functions and providing more reasonable learning curves for hyperparameter optimization. The paper (Arjovsky et al., 2017) describes the W-GAN and provides important information about the theory behind possibly implementing it in our model as a useful extension.

References

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein gan, 2017. URL https://arxiv.org/abs/1701.07875.

Gatys, Leon A, Ecker, Alexander S, and Bethge, Matthias. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Karras, Tero, Laine, Samuli, and Aila, Timo. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1812.04948, December 2018.

Kumar, Pawan and Goel, Nikita. Image resolution enhancement using convolutional autoencoders. pp. 8259, 11 2020. doi: 10.3390/ecsa-7-08259.

Liu, Zhi-Song, Kalogeiton, Vicky, and Cani, Marie-Paule. Multiple Style Transfer via Variational AutoEncoder. *arXiv e-prints*, art. arXiv:2110.07375, October 2021.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv e-prints*, art. arXiv:1703.10593, March 2017.