Training Conditional Adversarial Networks in One Stage

Aadarsh Sreekumar
Queen's University
Kingston, Canada
aadarsh.sreekumar@queensu.ca

Siam Shibly Antar
Queen's University
Kingston, Canada
23frl@queensu.ca

Spencer Keene
Queen's University
Kingston, Canada
spencer.keene@queensu.ca

1 INTRODUCTION

Generative Adversarial Networks (GANs), are a popular choice for image generation and transformation tasks. The adversarial nature of the generator and the discriminator produces images that are visually appealing and in many cases indistinguishable from real images. Image translation tasks such as black-and-white image colorization were traditionally done using convolutional neural networks (CNN). CNNs work by minimizing a loss function during training that poses a limitation for generating sharp and realistic images without extreme complexity. GANs or even conditional-GANs (cGANs) can work on a high-level goal such as generating high-quality images almost indistinguishable from reality [6]. This is possible due to their adversarial nature where the discriminator keeps on critiquing the generator until an image of acceptable quality is generated. We intend to implement the novel one-stage training method for GANs [14] and evaluate its application to a more trivial task such as image colorization. We are expecting to see a 1.5 times acceleration [14] and a similar improvement to the overall model performance too. Suppose the one-stage training methodology can be successfully applied to a use case such as image colorization. In that case, it can be extrapolated to be used in more complex generative problems [6] and even in problems not currently solved using traditional GANs. Thus, one stage could be a widely accepted new methodology for training GANs in the field of deep learning.

In our project, we aim to implement the novel one-stage training method for GANs and apply it to a more common problem solved by GANs which is image colorization. We expect substantial acceleration and an improvement in the overall model performance. For this project, we shall be following the pattern commonly followed in a machine learning pipeline. The pattern includes data collection and preparation, model training, model evaluation and analysis, validation, and finally deployment. In this proposed method of training GANs in only one stage [14], we categorize the GANs into two classes based on the adversarial losses of the generator and the discriminator, namely symmetric and Asymmetric GANs. A novel gradient decomposition method [14] is then used to unify the two classes, thereby training both classes in one stage and improving training efficiency.

Once this is implemented in place of a traditional cGAN to colorize black and white images, we shall also computationally analyze the performance of the new implementation and test whether the 1.5 times acceleration claim [14] holds true for our use case too. If we can achieve this, we shall also investigate if we can implement this novel GAN training to other problems that are being solved by traditional GANs today. The datasets we are using are as follows: ImageNet, COCO, MNIST, LSUN Churches, CIFAR10, and CelebA, which are the same datasets that were used in the existing works [14] [6] [12] and our references. These datasets contain a very high number of images which makes it a great resource for image processing and deep learning workloads. We will be converting the RGB images present in these datasets into the LAB color space and only provide the L (Luminance) channel to the cGAN network to train. For our project and computational feasibility, we shall be using a subset of these datasets for our training, considering our computational limits as covered in §4. Finally, in this project, the implementation of the novel one-stage training technique for image colorization using cGANs has been achieved and shows better efficiency and great promise in this domain, details of which will be covered in the later sections of this paper.

2 LITERATURE REVIEW

A recent technique in the field of generative modeling is the use of Generative Adversarial Networks (GANs), which were first published in [6]. GANs have an interesting approach, where they coordinate two simpler models together in an adversarial manner. These two models are referred to as the generative model and the discriminative model. The primary function of the generative model is to produce synthetic data instances that closely resemble an existing dataset. On the other hand, the primary function of the discriminative model is to estimate the probability that which these synthetic data instances output by the generative model can be indistinguishably associated with the original dataset. By pairing these models against each other, the generative model is essentially learning a method to optimize the likelihood of the discriminative model making a mistake by classifying its output as real data. This adversarial approach is critical to a GAN's ability to produce impressively realistic synthetic data.

Conditional GANs (cGANs) are an exciting extension of GANs that were investigated as a general-purpose solution for image-to-image translation tasks [6] [2]. Traditional GANs use only noise as input but cGANs introduce an additional input, conditioning the generative model's output on this additional information, thus increasing their utility and applicability. For the purposes of image-to-image translation, this additional input is the starting image, which allows the cGAN to generate a new image based on the first one. Some examples of common image-to-image translation tasks are converting black and white images to colored images, converting sketches to realistic photographs, and converting satellite images to maps or vice-versa. [6]

Traditional GANs have a disadvantage in their training process, despite their remarkable capabilities. This method involves alternating training stages in which the generative and discriminative models take turns updating, making the training process time-consuming. A revolutionary approach presented in [14] consolidates the training of both models into a single stage, considerably speeding up the training process. Empirical results suggest that this one-stage training strategy can reduce training times by about 1.5 times across a variety of datasets. It is worth noting that, while this streamlined training strategy has proven beneficial in the setting of traditional GANs, its relevance to cGANs has yet to be investigated. It does, however, show significant potential for extending similar benefits to conditional GANs, representing an intriguing option for further research and exploration.

3 METHODOLOGY

In this section, we shall delve into the details of the deep learning methods and models that we have used in our project. Particularly we shall look at both the existing conventional cGANs where the generator and the discriminator are updated in separate stages hence called the two-stage training method, which is the most commonly followed and the most famous methodology for training. We shall then also look at the novel single-stage trained GAN which boasts improved performance and reduced training times leading to better efficiency. The main focus of this project is to implement the single-stage training technique to image colorization cGANs.

3.1 Traditional Conditional GAN

Generative Adversarial Networks are generative models in the domain of deep learning that can capture a mapping from random noise to generate an output image where as conditional GANs widely known as cGANs learn a mapping from a pre-existing image along with some random noise to generate an output image finally. [5] The generator is trained to produce outputs that cannot be distinguished from the real images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's fake images from the real ones. To perform image colorization, the traditional GAN has been converted to a traditional cGAN to generate images from images and some noise instead of just from noise as mentioned above. Specifically, in this scenario, the generator takes the L-channel of the LAB colorspace image as its input and then proceeds to generate colorized images. The discriminator is trained on both the ground truth image as well as the generated images to determine if the image is fake or real. [6]

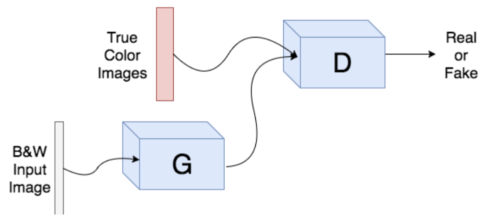

Figure 1: Conditional GAN Visualized

To mathematically represent the objective function for the conditional GAN, we can think of it as a min-max game. The objective of the generator is to minimize this loss function and the objective of the discriminator is to maximize the same function. We can observe this in action in upcoming sections where we shall analyze the training through data visualization. The Loss function can be represented as follows:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]$$

Interestingly, Previous works and approaches have discovered that the benefits of mixing more conventional losses such as L1 or L2 distance can lead to better results. In this context of image generation, L1 distance has shown great promise as it encourages less data loss and prevents blurring thereby leading to sharper results. Therefore our final objective function can be defined as the following.

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Moving into the architectural details, we can observe a concurrent theme across various solutions that the generator model often uses the U-NET model. The U-NET model can be visualized using figure 2, which illustrates only the inner three layers of the U-NET architecture whereas in reality there will be more layers in this network. Looking at the discriminator, we see that the model has been created by a combination of the Conv-BatchNorm-LeackyReLU layers by using a stacked variant of these models.

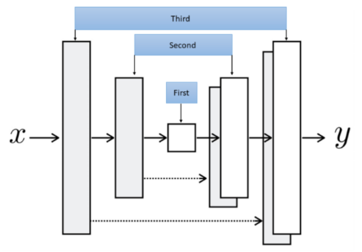

Figure 2: U-NET Model Architecture

Having a solid foundational understanding of the traditional cGAN model is helpful in optimizing this model and thereby leading to the novel invention of the single-stage trained conditional networks.

3.2 Single Stage Conditional GAN

The single-stage trained GAN works on a novel one-stage training scheme, called One-Stage GANs (OSGAN) [14], which can generalize to both Symmetric and Asymmetric GANs. The primary objective is to integrate the optimization for the generator and the discriminator during their forward inference and decompose their gradients during the back-propagation step to update them in one stage rather than do it in two stages as visualized in Figure 3.

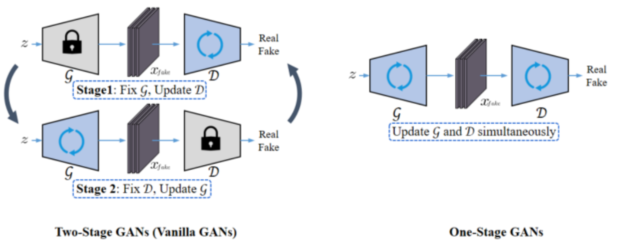
z
$\mathcal{G}$
x_{fake}
$\mathcal{D}$
Real
Fake
Stage1: Fix $\mathcal{G}$, Update $\mathcal{D}$
Stage 2: Fix $\mathcal{D}$, Update $\mathcal{G}$
Update $\mathcal{G}$ and $\mathcal{D}$ simultaneously
Two-Stage GANs (Vanilla GANs)
One-Stage GANs

Figure 3: OSGAN Compared to Traditional cGAN

The paper [14] mainly categorizes the models into two main types of GANs which are trained a bit uniquely with this method namely Symmetric and Asymmetric GANs. In Symmetric GANs, the generator and the discriminator losses contain the same loss term while in the case of asymmetric GANs, this is not the case as the losses contain different terms. For the Symmetric case, since the discriminant loss holds a term that is identical to the generator loss, we only need to compute the gradient of this term once and adopt it for both losses. In this way, the updates of the generator and discriminator may safely take place in one forward and backward step. The loss function for the Symmetric GAN where the logarithmic term is the same for both the generator and the discriminator is as follows:

$$\mathcal{L}_D = -\log D(x) - \log(1 - D(G(z))),$$
$$\mathcal{L}_G = \log(1 - D(G(z)))$$

When it comes to Training Asymmetric GANs it gets much trickier since we can no longer just copy the gradients derived from the discriminator to the generator as discussed above. Thus, the literature [14] looks into the composition of gradients and it is found that the gradients from the adversarial terms do indeed preserve their proportions within the total gradients from the previous layer all the way back to the first layer of the model. This interesting property of the losses enables us to mathematically convert these Asymmetric GANs into a synthetic form of Symmetric GANs and thereby perform the single stage-based training on them. Thus the loss function set for Asymmetric GANs is as follows:

$$\mathcal{L}_D = -\log D(x) - \log(1 - D(G(z))),$$
$$\mathcal{L}_G = -\log(D(G(z)))$$

Thus, to provide a more instructive visualization of this training methodology, let's use the following representation to grasp the training algorithm.

Therefore, the above algorithm captures the essence of the mathematical methodology we have covered in this section by listing the steps that are required to perform training using the single-stage technique.

Algorithm 1 One-Stage GAN Training Framework

```
Input: Training data $X = \{x_j\}_{j=1}^N$
Output: The parameters of generator $G$.
1:  Initialize the parameters of $G$ and $D$.
2:  for number of training iterations do
3:      Sample $z \sim \mathcal{N}(0, 1)$ to generate fake data $\tilde{x}$ by $G$;
4:      Sample real data $x$ from $X$;
5:      Feed $\tilde{x}$ and $x$ into $D$ to compute $\mathcal{L}_G$ and $\mathcal{L}_D$;
6:      Compute $\nabla_{\tilde{x}}\mathcal{L}_G$ and $\nabla_{\tilde{x}}\mathcal{L}_D$ to obtain $\gamma$ by Eq. 7;
7:      Compute $\mathcal{L}_D^{ins}$ by Eq. 9;
8:      Back propagate $\nabla_{\tilde{x}}\mathcal{L}_D^{ins}$ to obtain $\nabla_{\tilde{x}}\mathcal{L}_D^{ins}$;
9:      Obtain $\nabla_{\tilde{x}}\mathcal{L}_G^{ins} = \gamma \cdot \nabla_{\tilde{x}}\mathcal{L}_D^{ins}$ by Eq. 7;
10:     Update $G$ and $D$ simultaneously using Adam.
11: end for
```

4 DATA COLLECTION AND PREPROCESSING

The nature of the data we have used in this project is all images since our work is aimed at colorizing black and white images using cGANs. Thus, we have reviewed our reference papers and existing works and we have observed that the ImageNet dataset is a very popular dataset for deep learning workloads involving image processing and image generation. Also, the ImageNet, COCO, MNIST, LSUN Churches, CIFAR10, and CelebA datasets also showed popularity amongst researchers in the domain. Considering our computational limitations and technical feasibility, we have gone ahead and extracted a subset of 4000 images from the above-mentioned datasets and other sources like Kaggle for example to compile our own dataset which we have used for training. While selecting this dataset, we have ensured to select a good sample from multiple contexts so as to prevent any bias towards a certain type of picture when training the model. Finally, in our single-stage trained cGAN for colorization, we shall be using bright and vibrant animated images for inference to showcase the colorization ability of the GAN on images and this has also proven to work better with our limited training set.

The model used in this project works best using an image of size $256x256$. We have chosen this size as it is compact enough to train and run using the resources available at hand without compromising a lot on details or spatial features. Therefore, before we can begin training, we have performed a necessary step of downsampling all images in our dataset to the $256x256$ resolution so that it can fit into our models effortlessly. Next, we shall segment the channels present in our RGB image so that we are able to convert it into the LAB color space. The luminance channel (L) from the LAB color space will be used as input to the generator and the discriminator present in the model. The remaining color channels (AB) will be used to calculate the error concerning the output from the generator.

These steps will be carried out by the data generator which has the responsibility of providing each batch into the model and performing the following tasks. Firstly, we read the RGB values of the images in the dataset. Now we shall flip some bits randomly in the horizontal direction to perform a simple way of data augmentation which acts as a regularization technique while training the model. Next, we shall convert the RGB image to LAB color space for the intents and purposes as stated above. Next, we normalize the L-channel values from [0,100] to [0,1] for better training and normalized data. Next, we shall also normalize the values of the AB-channel values from $[-128, 128]$ to $[-1, 1]$ which will be used in calculating the error in training. Finally, we can convert the Numpy matrix with the value into PyTorch tensors thereby enabling us to separate the L & AB channels and pass it into the model for computation.

5 IMPLEMENTATION DETAILS

We have implemented our project using PyTorch since the majority of deep learning and machine learning projects in academia are written in PyTorch thereby making it easier to implement ideas proposed in papers. We have had to use personal systems with older consumer-grade Nvidia CUDA GPUs for our implementation and we can guarantee better results if we can use server-grade or professional-grade high-performance computing resources for further future training. In our implementation of the published work, we have observed that the single-stage training, although proposed to be efficient in training, still requires enormous computational resources for training large datasets, although it is lower than the traditional computational requirement of an ordinary cGAN.

We can thereby state that this hurdle has made our effort of implementation quite cumbersome since we don't possess the adequate computational resources required. Specifically, our extent of exploration and length of training as well as the data used for training the single-stage trained GAN example was limited due to the unavailability of high-performance computing. However, for our use case, we were able to generate satisfactory results using the resources available on hand.

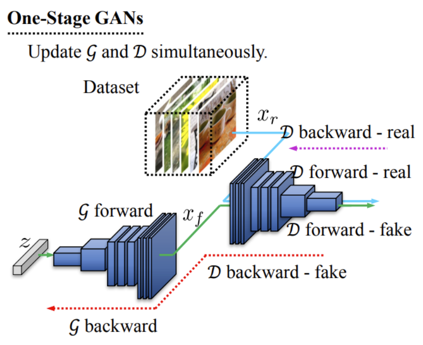
One-Stage GANs
Update $\mathcal{G}$ and $\mathcal{D}$ simultaneously.
Dataset
x_r
$\mathcal{D}$ backward - real
$\mathcal{D}$ forward - real
$\mathcal{G}$ forward
x_f
$\mathcal{D}$ forward - fake
z
$\mathcal{D}$ backward - fake
$\mathcal{G}$ backward

Figure 4: Single Stage GAN Layers

The single-stage trained GAN uses the Adam optimizer for both the generator and the discriminator and group normalization is used instead of batch normalization. We have also understood that, for training cGANs, a pre-trained model can ensure stability in training. Our objective is to build onto the existing models thereby easing training and improving accuracy. We have also inferred that the discriminator should execute more times than the generator for optimal results. Thus, we have performed transfer learning in essence where we used a pre-trained generator and discriminator model and further trained on it using our dataset. We have also had to scale down our images to 256x256 pixels each so that the images could fit into the model without any expensive sampling techniques. Upon training, we also noticed that using higher batch sizes drastically brought down performance and often caused the GPU memory to be completely exhausted even leading to CUDA-NN errors during training. To resolve this issue we have had to use lower batch sizes so that our training could continue smoothly.

6 EXPERIMENTAL DESIGN

In this project where we have to evaluate the image generated by the model, we can use multiple performance metrics to evaluate the effectiveness of the model. The major performance metrics which are used in this domain are as follows:

- Image Evaluation Through Human Observation
- Overall Model Training Time
- Fréchet Inception Distance
- Kernel Inception Distance

6.1 Image Evaluation Through Human Observation

Observing the image that has been generated by the GAN and comparing it with the black and white image to understand the context and the object present in the image can help us understand and thereby evaluate how well the model can colorize an image. After all, the colorization of each image greatly depends on the type of image, the objects present in the image, and the context of the image which was passed as an input. We find that this metric is the most suitable metric to evaluate the performance of the model.

6.2 Overall Model Training Time

The overall time taken to train the model along with the time taken for inference is also another good metric to evaluate the performance and effectiveness of a model. Since the single-stage training method is supposed to reduce the overall training time of the model, it will be an interesting metric to consider to evaluate the model.

6.3 Fréchet Inception Distance

The Fréchet Inception Distance is a metric used to evaluate the quality of generated images in generative models, particularly in the context of generative adversarial networks (GANs). It quantifies the similarity between a pair of images by comparing them.

The formula for Fréchet Inception Distance (FID) involves computing the distance between multivariate Gaussian distributions represented by their mean vectors and covariance matrices.

$$\text{FID} = \|\mu_1 - \mu_2\|^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1\Sigma_2})$$

Where:

FID	: Frechet Inception Distance
$\|\cdot\|^2$	: Squared Euclidean distance between mean vectors
$\text{Tr}(\cdot)$	: Trace of a matrix
μ_1, μ_2	: Mean vectors from real and generated images
Σ_1, Σ_2	: Covariance matrices from real and generated images

6.4 Kernel Inception Distance

Kernel Inception Distance (KID) is a metric used to measure the similarity between two distributions of features extracted from real and generated images. It's an extension of the Fréchet Inception Distance (FID) that aims to capture the difference in distributions using kernel methods.

$$\text{KID} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}k(x_i,x_j)-\frac{2}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{m}k(x_i,y_j)+\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}k(y_i,y_j)$$

Where:

KID : Kernel Inception Distance

n : Number of samples in the real image feature distribution

m : Number of samples in the generated image feature distribution

$k(\cdot,\cdot)$: Kernel function (e.g., Gaussian kernel)

x_i, y_i : Feature vectors from real and generated images

6.5 Best Metrics for the Project

In our project, we shall be mainly implementing the image evaluation through human observation as the primary metric as it provides the most detail when it comes to a task such as image colorization, which vastly depends on the context of the image which can't be mathematically implied always. We shall use the Fréchet Inception Distance metric so that we have a quantifiable metric for comparison and evaluation. Finally, we shall also compare the training time and performance of the model along with the above-mentioned metrics in §7.

7 RESULTS AND ANALYSIS

In this section, we shall present and analyze the results we have obtained in this project. We shall use graphical representation such as charts to visualize training and we can also use the outputs generated to visualize and interpret the effectiveness of our models. Going ahead, we shall segment this section into two parts, one for the traditional GAN and the next one for the newly implemented single-stage trained GAN for image colorization.

7.1 Colorization using Traditional cGAN

Traditional cGANs or the conditional generative adversarial networks that use the well-established two-stage model to train is the most common model that is currently used in the domain of image generation tasks, importantly image colorization which is our topic of focus. In this method, our model architecture is the same as we have discussed in the above sections §3. We have gone ahead and used our specially created dataset to perform training. We have used the Queens University 'L1NNA' high-performance cluster for training this specific model as it is extremely resource-intensive. We can proceed to the evaluation phase by recalling the evaluation criteria we had set in §6. We shall primarily shine a light on the most important evaluation criteria which is the judgment of colorization accuracy using human observation and also the training time. Before we jump into further analysis, let's look at the training time it took for the conventional cGAN against our dataset.

*Training Time for Conventional cGAN: ~**7 Hours & 18 Minutes***

Considering our computational limitations and the resources available at hand, this is a satisfactory time frame in which our training has concluded. Note that, we had used a pre-trained GAN model to continue our training, as we are performing a variant of transfer learning in our implementation.

Moving onto analyzing the training phase, let's observe and understand the most common plot (Figure 5) which is the convergence curve of the Generator and the Discriminator over Iterations.

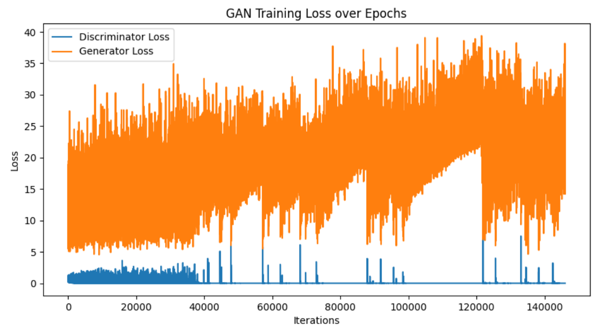

Figure 5: cGAN Training - Loss vs Iterations

As discussed in the above sections, the generator The goal of training is to keep the discriminator accuracy near 100% which means that the discriminator loss should tend towards minima or zero and make sure that the generator loss doesn't drop to very low levels. This is because if the generator loss drops to low levels then it can be because the generator is fooling the discriminator with bad colorization which leads to bad training of the model.

We can observe that the discriminator loss is relatively low and stable compared to the generator loss, which suggests that the discriminator is performing well at distinguishing between real and generated samples whereas the generator loss is higher and the training does look noisy which could imply that the generator is not being able to effectively fool the discriminator as its pretty good. We can also observe that the graph compares against iterations instead of epochs. The actual value of epochs used in this training is around 140, but still, we aren't close to a visible point of convergence. This means that to obtain satisfactory results the training needs to continue beyond the range set by us, which would definitely require higher computational resources.

Now let's look at the colorization output generated by this model. We shall compare the models' ability to effectively colorize the image from black and white and evaluate its accuracy from a human observational point of view. The reason why we have used such images for our inference is because the original model from which we have performed transfer learning trained on such types of images from its dataset.

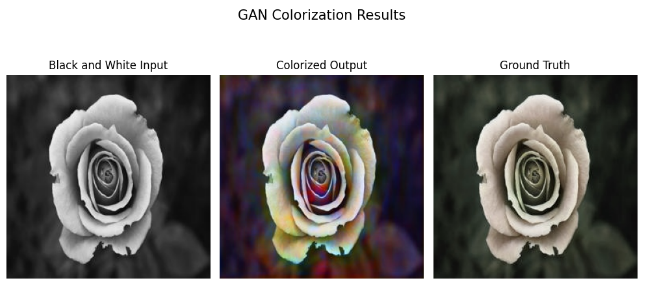

Figure 6: Image Colorization - Rose Image

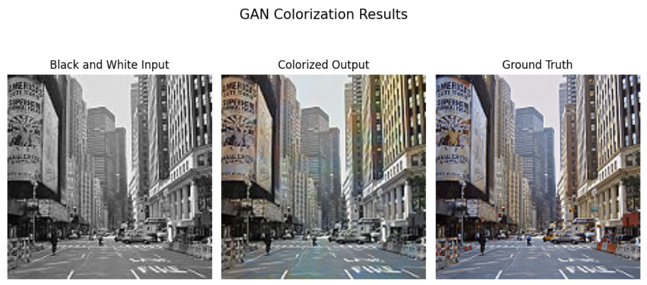

Figure 7: Image Colorization - Street Image

We shall only use these two outputs for our further analysis as we believe that this can capture the essence of the entire inference dataset we used after training. Upon a primary glance, we can observe a significant amount of fringing in these images, especially in Figure 6. This can be due to the dataset used for training or it is most likely the GANs inability to successfully color complex images with multiple layers and ambiguity. For example, roses can be of various colors and the model has to decide on a general solution when presented with this image. However, the street scenery image looks to be colored with good accuracy and minimal loss of information. Upon closer inspection, we can see that the saturation is not up to the level of the ground truth image and there are portions of the image which are incorrectly colored but all in all this was a good colorization result.

7.2 Colorization using Single Stage cGAN

Training cGANs using the one-stage method as detailed in the former sections is most likely going to reduce training times and improve performance thereby enabling an improvement in the overall accuracy and effectiveness of the model at its task, which in our case is image colorization. Since one-stage training adopts the synchronous update for both the generator & the discriminator, it can significantly avoid the early gradient saturation in the two-stage strategy thereby achieving a higher likelihood of faster convergence when compared to §7.1. To attest to these claims the authors of the original OSGAN paper [14] have included some interesting graphics (Figure 8) in their work which establish the fact that one-stage training is indeed better than two-stage training technique.

As we can see in Figure 8, two-stage training does help in improving the overall efficiency of the model. Now, we can observe

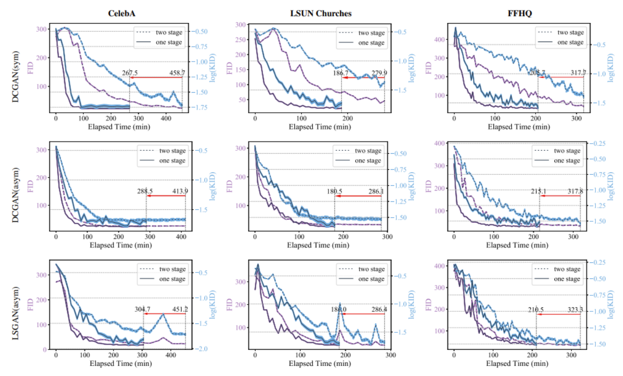

Figure 8: Comparison of Training Efficiency between OS-GANs & TSGANs

the results collected in our project to see if our findings correlate with the intended results of single-stage training. Before jumping into the exciting part of visual analysis, let's also look at the training time performance evaluation metric and compare it with the previous section.

*Training Time for Single Stage cGAN: ~**4 Hours***

This is significantly lower than the training time observed with the traditional cGAN and we can confidently state that this model outperforms single-stage training. To put it into perspective, we can also state that one-stage training cGANs for image colorization gives a boost of ~1.6X in our study.

Let us now visualize and analyze (Figure 9) the convergence curve which is the generator and discriminator loss against the number of epochs for the single-stage trained colorization cGAN.

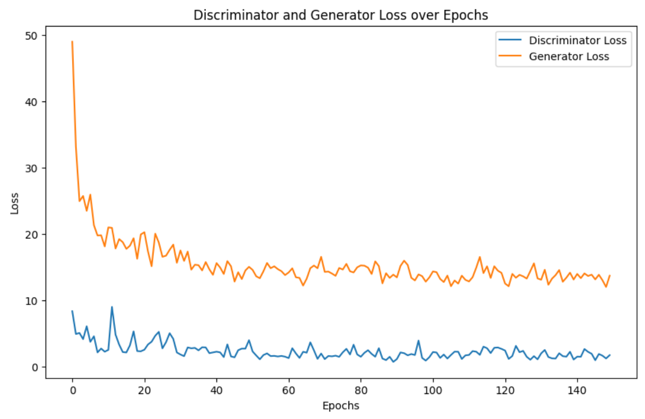

Figure 9: Single Stage Colorization GAN Training - Loss vs Epochs

The graph from our training shows the loss curves for both the discriminator and generator in our single-stage trained cGAN during training across epochs. Initially, both losses start high, with the discriminator loss showing a sharp decline in the first few epochs before stabilizing. The generator loss decreases more gradually. As training progresses, the discriminator loss fluctuates slightly but remains relatively low, indicating that it's effectively distinguishing between real and fake data. The generator loss, after its initial decline, maintains a consistent level, with some variability but there is no clear trend upwards or downwards in scope, however, the training is stable. This could suggest that the generator is improving at creating believable images, but it's not necessarily gaining ground on the discriminator. The relative stability of both curves towards the end of training may indicate that the model is reaching a point of convergence, where neither the generator nor the discriminator is significantly outperforming the other. This could also mean that with higher computational resources on hand, the training can be further extended so as to minimize losses further and get a refined colorization model.

Upon comparing this result with the graph from the traditional cGAN we can observe that the single-stage colorization model is outperforming with a significant margin as it boasts better stability, lesser noise, improved training and learning dynamics, and an end goal towards convergence. It's important to keep in mind that this result can be an oversimplification due to our trivial use case and might or might not exhibit varied behavior in other scenarios.

We can now analyze the output generated by our newly trained single-stage model, We shall compare the models' ability to effectively colorize the image from black and white and evaluate its accuracy from a human observational point of view as performed above. Please note that for this use case, we will be illustrating our models' ability to colorize images by using animated images which can show the potential for coloring multiple regions with a variety of colors.

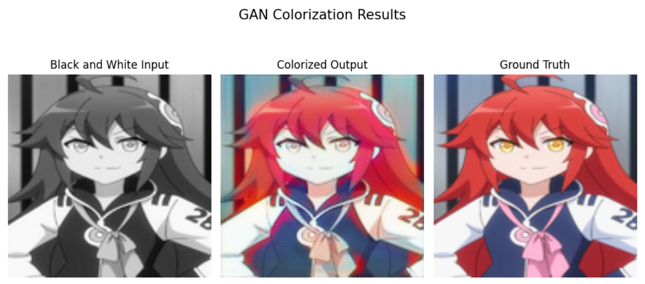

Figure 10: Single Stage Image Colorization Output 1

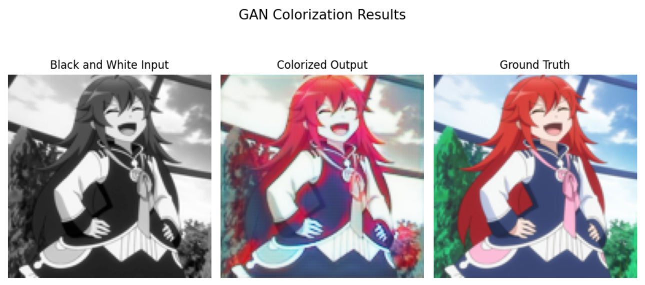

Figure 11: Single Stage Image Colorization Output 2

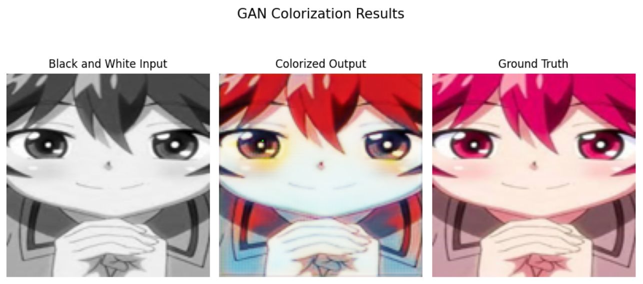

Figure 12: Single Stage Image Colorization Output 3

The results which are generated by the single-stage cGAN look much better when compared to the results generated by the traditional cGAN as exhibited in the above sections. We can notice that the colors of the generated output are very close to the actual ground truth. However, we do notice some blurring around the edges in figures 10 & 11, but these minor defects can be ignored as they can be fixed with sharpening in post-processing or some other detail preserving technique can be applied. Looking closely at figure 12 we can also see some green pixels indicating areas of uncertainty or noise which hopefully get fixed as training progresses beyond our limit of 150 epochs set due to the computational constraint we face. To quantify the improvements, the single-stage trained outputs exhibit lower Fréchet distance to the ground truth labels when compared to the outputs generated by the traditional cGAN. The saturation and colorization accuracy of the outputs generated are also remarkably good, although one hundred percent accuracy hasn't been achieved in any use case. Therefore we can confidently establish the fact that single-stage training of GANs do offer significant performance improvements and definitely should be the way to go forward in this domain.

8 DISCUSSION

In this project, the performance of traditional two-stage cGANs and the novel single-stage cGANs for image colorization has been analyzed. Traditional cGANs are a common occurrence in image generation workloads and are resource-intensive thereby leading the model to be trained on the university's GPU computing cluster, completing in around 7 hours & 18 Minutes. This approach along with utilizing a pre-trained model for transfer learning is the common practice for training such models. The Evaluation criteria focused on colorization accuracy assessed by human observation and overall training duration. The discriminator's low and stable loss versus the higher, more volatile generator loss suggested effective real versus fake distinction but also indicated potential training instabilities or insufficient generator complexity.

The new single-stage trained cGAN, leveraging simultaneous updates for the generator and discriminator, aims to expedite training and enhance model accuracy. The model completed training in about 4 hours, showing an excellent improvement in efficiency, a boost of around 1.6X. This method's strength lies in its training efficiency, substantially reducing computational time without sacrificing output quality. The single-stage cGAN not only accelerates the training process but also seems to stabilize the adversarial training dynamics, which is a notable advancement given the typically volatile nature of GAN training. The training loss curves revealed a more stable and promising trend toward convergence, hinting at the model's improved capability to generate believable images. However, further training with more computational resources might yield even more refined results. Comparatively, the single-stage model demonstrated superior performance with better stability, less noise, and a clearer direction toward convergence. While this indicates an oversimplification due to the project's specific use case, the improved results suggest single-stage training as a favorable method in the domain.

Furthermore, output analysis from the single-stage cGAN showed more accurate colorization, with minor blurring and noise that could be addressed with additional epochs or post-processing techniques. The single-stage outputs also registered a lower Fréchet distance to the ground truth compared to the two-stage model, denoting a closer match to real images. Although not perfectly accurate, the single-stage model's outputs were significantly better than the traditional cGAN's, establishing the efficacy of single-stage GAN training in image colorization tasks.

Therefore, the insights we have gained from this project could guide future research in GAN architectures, particularly in exploring more efficient ways to train deep learning models for complex image processing tasks such as using single-stage trained GANs, potentially broadening their applicability and accessibility.

9 CONTRIBUTIONS

Generative adversarial networks have been the popular choice in computational problems requiring data generation, especially in the realm of images and videos. Improving on the current performance and training effort of GANs will have a huge impact on the way GANs are implemented across the domain and will make GANs a very attractive solution for problems not solved by GANs today. Such technological advancements that improve the technical feasibility of using these models will motivate more research and development in this domain which will lead to advancements in deep learning and machine learning.

10 CONCLUSION

In Conclusion, we have managed to implement the One Stage Training technique for Conditional Generative Adversarial Networks (GCN) and performed training and inference using our systems. We have also managed to train and implement the conventional two-stage based cGAN for colorizing images. Finally, we have also managed to implement the one-stage training methodology for the aforementioned task of colorizing black-and-white images and observed improved training times and performance gains. We also were able to generate more photo-realistic colorization outputs with our newer model. We believe that the accuracy and the overall polish of the model could be further improved if we had access to powerful GPU clusters for further training, thus reaching a final conclusion that the one-stage method however improves training time and exhibits better performance still needs to be trained well using appropriate data to match the effectiveness of current solutions. In terms of future work or research, we can extend this project to other generative tasks which use cGANs such as Labels to Facade, Day to Night, Maps to Aerial Image, Sketch to Photo, Edges to Photo, Deep-Fakes, Semantic Labels to Scenes, Thermal Images to Colorized Photos [6] and observe better training times and improved performance.

REFERENCES

[1] Andrew Brock et al. 2019. BigGAN: Large scale generative adversarial networks. In *International Conference on Learning Representations.*

[2] Rashi Dhir, Meghna Ashok, Shilpa Gite, and Ketan Kotecha. 2021. Automatic Image Colorization Using GANs. *Springer eBooks* (Jan 2021), 15–26. https://doi.org/10.1007/978-981-16-0708-0_2

[3] Aroosh Fatima, Wajahat Hussain, and Shahzad Rasool. 2020. Grey is the new RGB: How good is GAN-based image colorization for image compression? *Multimedia Tools and Applications* (Sep 2020). https://doi.org/10.1007/s11042-020-09861-y

[4] Leon A. Gatys et al. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition.*

[5] Ian Goodfellow et al. 2014. Generative adversarial nets. In *Advances in neural information processing systems.*

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jul 2017). https://doi.org/10.1109/cvpr.2017.632

[7] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR).*

[8] Christian Ledig et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition.*

[9] Chuan Li and Michael Wand. 2016. Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1604.04382

[10] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least Squares Generative Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct 2017). https://doi.org/10.1109/iccv.2017.304

[11] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[12] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. 2018. Image Colorization with Generative Adversarial Networks. *arXiv:1803.05400 [cs]* 10945 (2018), 85–94. https://doi.org/10.1007/978-3-319-94544-6_9

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science* 9351 (2015), 234–241.

[14] Chengchao Shen, Youtan Yin, Xinchao Wang, Xubin Li, Jie Song, and Mingli Song. 2021. Training Generative Adversarial Networks in One Stage. *arXiv (Cornell University)* (Jun 2021). https://doi.org/10.1109/cvpr46437.2021.00336

[15] Christian Szegedy et al. 2014. Intriguing properties of neural networks. *Proceedings of the International Conference on Learning Representations (ICLR)* (2014).

[16] Richard Zhang et al. 2018. *Image colorization: a survey and a new example-based method.* Vol. 24. 2907–2926. https://doi.org/10.1109/tvcg.2017.2744780

[17] Jun-Yan Zhu et al. 2017. P2P: A large-scale dataset for photographic image synthesis. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017).

[18] Jun-Yan Zhu et al. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision.*