1 Introduction

In the groundbreaking 2014 paper by Goodfellow et. al, **generative adversarial networks**, otherwise known as GANs, were introduced to the greater deep learning community [1]. As such, they have been a field of considerable research since their inception. Revolving around the idea of training a Discriminator D and Generator G in tandem, these models have been both studied and utilized for their effective generation of fake data, with said generation aiming to replicate real data to a high degree of accuracy. Of note, and of further discussion in this report, is the training mechanism for both D and G - which is typically a *purely* adversarial approach.

In this adversarial mechanism, the primary objective is to reward the Generator for creating data that the Discriminator cannot effectively classify as real or fake. In starkly opposing contrast, the Discriminator is rewarded for the correct identification of real and fake data. This back-and-forth procedure, if convergent, is meant to lead to realistic data generation.

Saying this, the model itself is not without its flaws. Mode collapse is of utmost notoriety, as it implies that the Generator has fooled the Discriminator to a point where outputs need not be diversified, or even realistic. This leads to practically identical outputs across the board. In addition, the assumption of convergence does not necessarily hold. Hence, if non-convergent, the data generated by the Generator will not be realistic, going against the point of the architecture itself. Finally, the issue of vanishing gradients, wherein the Discriminator is "too good," will also lead to overall poor performance.

In this report, we tackle a novel approach - introducing a reward/punishment mechanism with feedback loops, adding a *semi-cooperative* aspect to the generally adversarial game played between the Generator and the Discriminator. The goal is to consistently maintain a high quality of generated data, while also attempting to prevent the aforementioned issues prevalent in the training process from arising, with an overall goal of training stability.

We begin with a brief overview of the conventional concepts associated with generative adversarial networks, as well as the common challenges associated with them. We will then examine the intuitions associated with our reward/punishment mechanism, as well as the results associated with its implementation in practice.

2 Generative Adversarial Networks - The General Case

2.1 Idea and Structure

In any and all generative adversarial networks, of interest is the training of a Generator network to produce real-looking data. To facilitate the training of this Generator, a Discriminator network is defined, to determine whether or not the data generated by the Generator is passable as real data (hence, if the Discriminator can effectively discern real from fake data, the Generator clearly has some room for improvement). From the nature of the structure alone, it is clear that the Generator and Discriminator are actively working against one another.

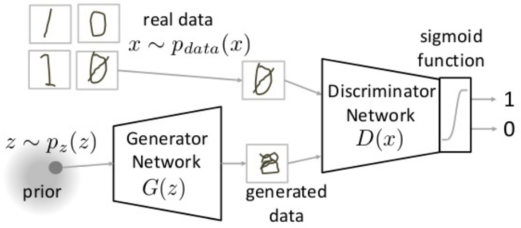


Figure 1: The General Structure of a GAN (c. Mark Chang)

2.2 The Adversarial Minimax Game

Consider the real data, x, being generated via its own distribution (i.e. $x \sim p_{data}(x)$, while the fake data, z, is also generated via its own distribution ($z \sim p_z(z)$). The typical GAN approach revolves around the adversarial game played between the Generator, G, and the discriminator, D. This game is referred to as the **minimax game**, and is defined as follows:

$$\min_G \max_D \; V(D, G)$$

where

$$V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log (1 - D(G(z)))]$$

In essence, the training of the G network is meant to maximize the likelihood of the D network making a mistake. Upon further optimizing the objective function, supposing that the Discriminator $D_{G^*}(x)$ is optimal, the optimal generator G^* will be such that $p_{data}(x) = p_z(z)$. So, the generator produces data that appears as if it comes from the true distribution. When this is the case, the Discriminator effectively makes a guess on whether or not the data is fake (i.e. $D_{G^*}(x) = \frac{1}{2}$). This is the goal when training a GAN, however, we hope to see whether the semi-competitive approach outlined can help in producing consistently better output than that of the conventional GAN structure.

3 The Proposed Approach - Reward/Punishment

3.1 Related Works

As aforementioned, the training process for GANs is primarily adversarial. There are many different papers outlining different games played for GANs, but at that, these games are still adversarial. Zheng et. al introduced a conventional reward mechanism to the adversarial game, accomplished by incorporating a rewarding term to the objective function of the Generator [2]. Separately, Chen et. al implemented the Wasserstein GAN under a reinforcement learning framework, applied by learning optimal policies and enforcing rewards as required [3]. Inspired by these papers, we are not only looking at the reward schema described prior, but also implementing a punishment. In addition, we are attempting to form a more robust relationship between the Generator and the Discriminator throughout the training process.

Our model introduces a *"semi-cooperative"* system, in which the Generator/Discriminator system is working together in a sense. We begin with an explanation of the procedure itself.

3.2 The Reward/Punishment Mechanism

As opposed to the traditional minimax game, we propose adding a "semi-cooperative" framework, wherein the Discriminator and Generator networks actively reap rewards based on their combined performance, and receives a punishment as well with a similar outline. Our implementation of this reward/punishment mechanism still utilizes the adversarial component of the minimax game in a sense, however, considers a phased approach, with a direct feedback mechanism employed for the Generator.

1. *The Punishment Phase*

   Similar to the conventional adversarial game, we begin the process by feeding the real data to the Discriminator, computing its loss, and then performing the same procedure for the fake data. However, we then apply a **punishment** based on the accuracy of the Discriminator output. Under this semi-cooperative framework, the Discriminator is partially responsible for the performance of the Generator. If the Discriminator has too high of a classification accuracy, the implication is that the Generator output is not realistic enough. In other words, the Discriminator is able to discriminate between real and fake data "too easily"- we punish the Discriminator, as it is not effectively providing assistance to the Generator. This assistance comes in the form of a feature-feedback mechanism, of which will be discussed further.

2. *The Reward Phase*

   We first train the Discriminator on a *subset* of the real data, and compute the loss based on this subset. As opposed to the punishment phase, wherein the real data and the fake data are considered separately, here we *combine* the remaining real data and the fake data prior to training the Discriminator. The reason for doing so is to avoid potential "cheating" under the semi-cooperative framework. If the Discriminator classifies the mixed data with high accuracy, it is then rewarded.

We start by considering the punishment phase for N epochs, where N is any positive integer. We then switch to considering a reward for epoch $(N+1)$, and revert back to the punishment mechanism afterwards.

We then shift attention to the Generator, where the feature-feedback mechanism is used. In the traditional GAN model, the Discriminator only outputs the final decision on whether or not a sample of data is real or fake. When using a feature-feedback mechanism, the Discriminator is modified so that the feature map of each output layer is also output. In the process of updating the Generator, we calculate the L_1-norm between the feature maps of real and fake data, and add it to the Generator's total loss.

Essentially, the system considers the reward/punishment guidelines outlined for the Discriminator, while the Discriminator also provides feedback to improve the Generator's learning - it can be thought of as a **student-teacher relationship**. The Discriminator being the teacher, and the Generator being the student. While the teacher's goal

is to be as strict as possible when evaluating student work, the teacher also provides detailed feedback regarding the student's work, so it can continue to improve. The punishment phase can be likened to the student getting feedback on homework before the big test - the student gets their work back, and can develop a deep understanding as to what the professor is looking for in the future. Thus, the reward phase is the test - wherein the adversity of the practicing phase pays off.

3.3 Intuition

The main reason this approach is suggested for the GAN structure, is due to their known issues with **training instability**. As mentioned prior, there are issues primarily with *mode collapse* and *vanishing gradients*. How these modifications could potentially alleviate these common issues is described below:

- *Mode Collapse*

  As stated prior, mode collapse occurs when the Generator begins to produce non-diverse output, failing to capture the desired diversity of the true data being emulated. In essence, the Generator finds a way to fool the Discriminator, without providing realistic output. By consistently applying rewards/punishments based on the Discriminator's ability, we aim to encourage the Generator to more accurately match the data by generating more diverse output. In addition, the Generator considers intermediate representations of the data in the feedback loop (understanding a "thought process" of sorts from the Discriminator), hence providing a further incentive to produce diversified output based on the feedback.

- *Vanishing Gradients*

  The issue of vanishing gradients arises when the Discriminator becomes effective too quickly, leading to a lack of gradient flow to the Generator, and thus a plateau in improvement for the Generator. By alternating between phases, we effectively prevent a rapid increase in Discriminator ability, leading to stable gradient flow throughout the training process.

In essence, the goal of introducing this system is to produce a better and more stable convergence for the GAN architecture, while also producing higher quality data. Next, this reward/punishment approach is implemented, and results are discussed.

4 Implementation & Results

We begin this section by providing a brief overview of the specific GAN architecture used, and the data being considered for evaluation.

4.1 Implementation

For the actual GAN architecture being used in experiments, we are using the PyTorch implementation of the *DCGAN (Deep Convolutional Generative Adversarial Network)* introduced by Radford et. al [4]. This is a popular variant of the regular GAN

architecture, wherein convolutional neural networks are used in both the Discriminator and Generator networks.

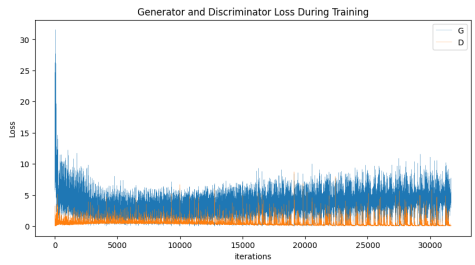
The data being used to train the model and generate results comes from the CelebA dataset, which contains around 226,000 images of celebrity faces - each of which being 3x178x218. There are roughly 10,177 unique celebrities present in the dataset, and our objective is to generate images that could pass as regular celebrities. All code used for these experiments is available through the LEARN dropbox.

As this functions as a brief overview into the structure being suggested, we consider two separate sets of values in the hyper-parameters of our model: we consider values that vary in their enforcement of the feedback, reward, and punishment considered. $S_1 = \{\lambda_{FB} : 0.6,\ \lambda_R : 0.25,\ \lambda_P : 0.1\}$, $S_2 = \{\lambda_{FB} : 0.4,\ \lambda_R : 0.1,\ \lambda_P : 0.02\}$, and $S_3 = \{\lambda_{FB} : 0.3,\ \lambda_R : 0.05,\ \lambda_P : 0.01\}$. Simply put, the higher any of these particular values are, the higher the magnitude of the applied feedback/punishment/reward. We consider $N = 4$ - the punishment phase being 4 epochs in duration.

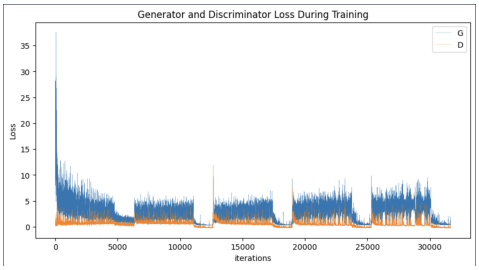
All other parameters were held constant throughout all 4 runs, with 20 epochs considered.

4.2 Results - Loss

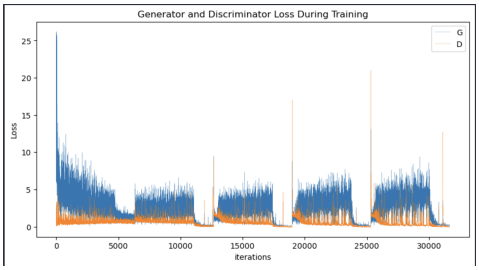
First, we will investigate how the proposed system alters the loss of the model.



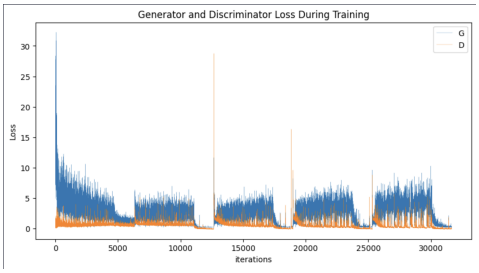
(a) Regular DCGAN Loss



(b) S_1 Loss



(c) S_2 Loss



(d) S_3 Loss

Figure 2: Loss of the models

As we can see, the effect of the implemented system is very clear throughout all hyper-parameter sets. However, there is a clear instability apparent in the S_2 and S_3 cases, with noticeable spikes in Discriminator loss. This is oddly counter-intuitive - as the associated values of λ_{FB}, λ_R, and λ_P **decrease**, the lack of overall stability in the training process appears to **increase**. So, when the novel training dynamics have less of an impact on the training itself, the system begins to fall into stages of considerable instability.

However, by the design of the reward system, these periods of rapid instability are quickly levelled out - while still troublesome, of course. Of note specifically is the timing of the rapid increases in both the S_2 and S_3 parameter sets. While the value of the parameters do not vary by a considerable margin (a small twofold increase in the λ_R and λ_P values, and a 0.1 increase in the value of λ_{FB} from S_3 to S_2), there is a noticeable difference in the occurrences of instability. In addition, there are evidently more spikes in the S_2 parameter set compared to the S_3 set.

Also of note is the magnitude of such spikes - it is rather strange for the highest Discriminator loss to occur midway through the training cycle, as indicated by the loss behaviour of the regular DCGAN architecture. These spikes could be attributed to periods of high Discriminator accuracy, wherein it is necessary by design of the system to punish a Discriminator that is too accurate too early in the training process.

This idea comes to a fault in the case of the S_2 parameter set, as the periods of instability come much later in the training cycle than those in the other parameter sets. So, the Discriminator may be over-punished immediately before training is finished, which in the case of the typical GAN approach, is illogical. As stated prior, the optimal Generator G^* will make the Discriminator $D_{G^*}(x) = \frac{1}{2}$, "guessing" whether or not the data is generated via the real or the fake data process. It appears that the Discriminator is still too accurate too deep into the training process, indicating a potential lack of stability.

Finally, what may be the most peculiar case is that of the S_1 parameter set - while the introduced dynamics are the most applied in a sense, the training is the most stable. This makes sense, as a significant departure from the original training dynamics of the GAN architecture must realistically be accompanied by a higher realization of the changes being made.

Now, it is time to investigate the actual outputs of each system, seeing whether or not the glaring instabilities factor into the realistic generation of images.

4.3 Results - Generation

We being by looking at an excerpt of the training data from the CelebA dataset,

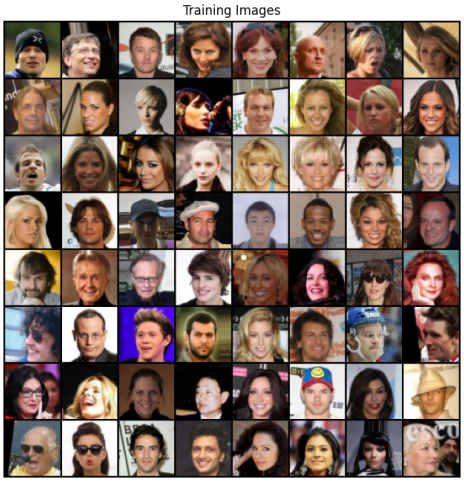
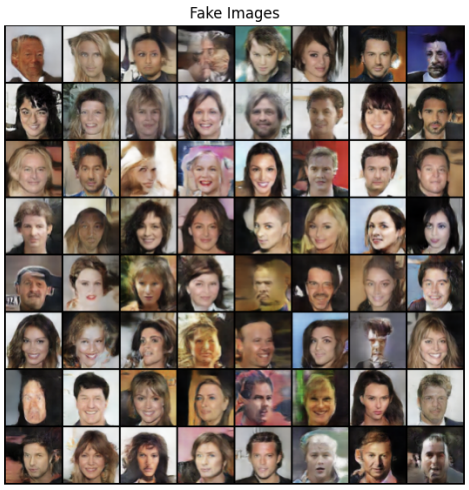
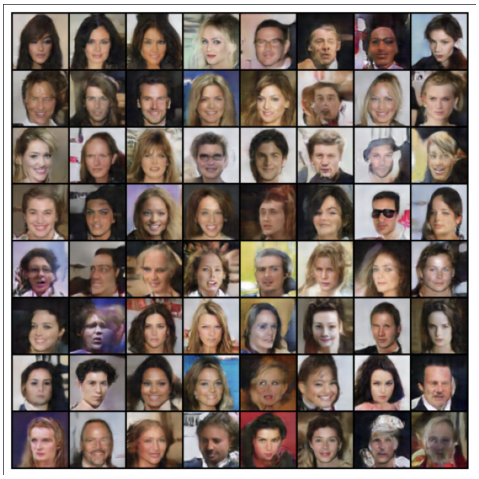


Figure 3: Example Images

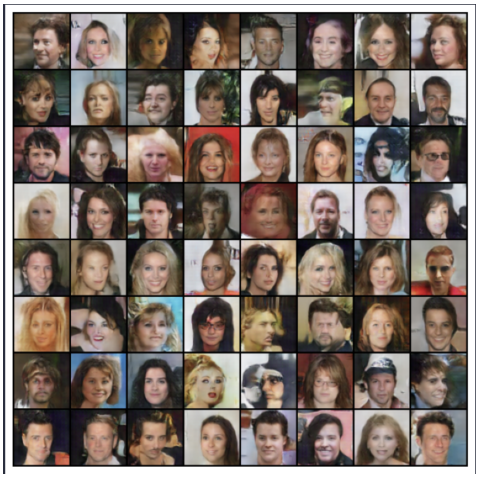
and immediately follow with the images generated via the original DCGAN, as well as our implemented system:



(a) Regular DCGAN Generations



(b) S_1 Generations



(c) S_2 Generations

(d) S_3 Generations

Figure 4: Generations by the models

Upon a close inspection, we see that none of these systems create realistic faces across the board. It is important to note that 20 epochs were used, so consistently realistic generation may come at the expense of time and higher computational resources. However, using a simple eye test, it appears that the outputs from S_2 and S_3 are more passable than those of S_1 and the original implementation. Taking a look at some of the "worst cases" exhibited in each generative process,

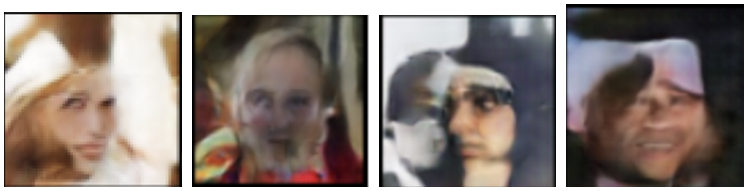


Figure 5: "Worst cases" of each model

We see that there are very similar cruxes in each generation - none of these methods are "perfect" in any shape or form. Due to an overall lack of quantitative metrics available for measuring how good a fake image is (as we could not implement something like Frechet Inception Distance due to computational constraints), we must simply judge how effective generation is by performing the eye test. This varies from person-to-person, so a strict evaluation will not be considered. It does appear that all cases of the reward/punishment system do not create as many unusual distortions as the original DCGAN, but once again this is very subjective.

While we cannot place a number on how well or how poorly generated images from our implementation look, we can say one thing for certain - the implemented reward/punishment system is **on par** with the original DCGAN with respect to image generation. So, while our implementation is admittedly rather low-level, such reward/punishment systems may be of considerable interest to those further researching generative adversarial networks.

5 Conclusion & Further Considerations

We have demonstrated a novel approach in the domain of the training and application of generative adversarial networks, using a reward/punishment system with feedback. This system is introduced to further control training stability (through prevention of mode collapse and vanishing gradients), while still generating believably real data.

While results are not promising immediately (due to the apparent instability of the Discriminator as a result of the reward/punishment mechanism), there are promising results. The developed process, through different magnitudes of the reward/punishment applications, seems to (at the very least) hold its own with regard to more general applications of the GAN architecture. There is certainly more work to be done outside the scope of this report, however. Some suggestions are as follows:

1. **Further optimization of** $\{\lambda_{FB},\ \lambda_R,\ \lambda_P\}$ - how much should the feedback, punishment, and reward of the model be considered in training?

2. **Further optimization of** N - How long, or short, should the punishment phase be? How much of a difference does this make?

3. **Investigate advanced rewards/punishments other than the accuracy of the Discriminator** - while we only considered accuracy as a metric in our experimentation, there are a host of other metrics that can be used to determine the magnitude of reward and punishment.

4. **Implement a rigorous metric to evaluate the performance of the models** - as stated prior, current methods for calculating how "accurate" generated images are can be computationally expensive and unreliable. So, of interest is an accessible metric for such a measurement.

While GANs are slowly losing interest in the public eye due to the advent of other generative systems such as Stable Diffusion and DALLE-3, we hope to have demonstrated that there are still plenty of different mechanisms that can be employed to further investigate GAN architectures.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.*Advances in neural information processing systems, 27*.

[2] Zheng, Changgang, Shufan Yang, Juan Parra-Ullauri, Antonio Garcia-Dominguez, and Nelly Bencomo.(2021). Reward-reinforced reinforcement learning for multi-agent systems. *arXiv preprint arXiv:2103.12192*.

[3] Chen, X., Yao, L., Wang, X., Sun, A., Zhang, W., & Sheng, Q. Z. (2021). Generative adversarial reward learning for generalized behavior tendency inference. *arXiv preprint arXiv:2105.00822*.

[4] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.