# 1 Introduction

We were very interested by the site thispersondoesnotexist.com. This page was created to showcase generated human faces using a General Adversarial Network (GANs). Every time you refresh the site, a new person is generated. These images are hyper realistic and are very diverse. We became inspired by this and found it fascinating how these generated faces seemed like real people, but were simply generated by an adversarial network. Since GANs are a few years dated, our goal was to use diffusion in realistic human face generation and see if we can achieve similar or better results.

While GAN's have been revolutionary for Generative AI, they face many challenges in training such as mode collapse, instability, and vanishing gradients during training. These limitations can hinder the quality and diversity of generated images. Diffusion models address the main pain points with GANs and in particular, the training process is much more reliable. Diffusion models employ a robust, iterative de noising process that ensures stable training and higher-quality outputs.

Our goal is to evaluate whether diffusion models can produce human face images that match or surpass the realism and diversity achieved by GAN-based methods like those used in thispersondoesnotexist.com.

# 2 Related Work

Generative models have evolved dramatically over the past few years. There can be considered two families: General Adversarial Networks (GANs) and Diffusion Models. GANs, take an approach using two neural networks- a generator and discriminator. The interplay between the two is critical to the success of GANs but has many shortfalls.

To give some background on the adversarial process that occurs between the generator and the discriminator, the generator maps a random noise vector from a prior distribution, such as a Gaussian. The objective of the generator is to generate data that tricks the discriminator in order to reduce the discriminator's ability to distinguish between real and fake samples. The Discriminator acts as a binary classifier, which estimates the probability that a given sample is real, from the distribution, or fake, from the generator. The role of the discriminator is to penalize the generator for producing unrealistic samples by distinguishing them from the real data.

GANs have demonstrated abilities to generate high-quality images, however, the adversarial dynamic presents several issues: mode collapse, training instability, non-convergence, vanishing and exploding gradients, and lack of theoretical

guarantees (Goodfellow et al., 2014).

Mode collapse occurs when the generator exploits patterns which consistently fool the discriminator which causes the generator to produce repetitive outputs. In the context of realistic human face generation, a GAN facing mode collapse issues would produce similar looking human faces, causing a lack of diversity in the produced output.

Training instability happens when the discriminator's feedback is too strong (overpowers the generator) or too weak (fails to provide useful gradients) which may cause divergence or oscillations in training. It is difficult to ensure convergence as it requires a careful balance between the generator and discriminator.

Non-Convergence may occur when GANs do not converge to the desired Nash equilibrium where $p_g = p_d$, which means the generator mimics the true data distribution ($p_d$) or the discriminator cannot distinguish real samples from fake generated data($p_g$). Instead, training may oscillate indefinitely or result in suboptimal equilibrium where neither the generator or discriminator improves.

Vanishing gradients occur early in training when the discriminator becomes too strong and easily distinguishes real from fake samples at a high confidence. This stalls the progress of the generator.

Exploding gradients occur due to poorly tuned learning rates or architectures which further destabilizes training.

GANs lack robust probabilistic formulations which makes their convergence guarantees weak, unlike diffusion models.

The above challenges significantly impact GANs' usability for projects requiring high diversity: due to mode collapse, fine details: instability during training and vanishing gradients, and scalability: more complex datasets makes training increasingly difficult.

Diffusion models address many of these issues.

Diffusion models avoid adversarial training completely, but utilizes probabilistic modeling in order to progressivity de-noise data.

The training process guarantees full coverage of the data distribution, solving mode collapse and instability (Yang et al., 2023).

Diffusion models also offer theoretical guarantees for convergence which makes them a more promising alternative for tasks like realistic human face generation.

# 3 Methodology

Generator Objective Function:

$$: \mathbb{E}_{z \sim p_z}[\log(D(G(z)))]$$

Figure 1: Generator Objective Function

The generator's objective is to maximize the discriminator's output for generated samples.

$z \sim p_z$: A random noise vector samples from a prior distribution, such as a Gaussian.

$G(z)$: The generator's output given a random noise vector z. Which transforms z into a sample in the data space.

$D(G(z))$: The discriminator's confidence that $G(z)$ is a real sample.

$log(D(G(z))$: Encourages the discriminator's output D(G(z)) to approach 1 for generated samples, which indicates the generator is fooling the discriminator.

This objective function is part of the adversarial training process where the generator attempts to maximize the discriminator's prediction for its generated samples, basically trying to make the discriminator believe the generated data is real.

The discriminator D(x) is trained to output 1 for real data (x from the real distribution $p_d$) or 0 for fake data (x from the generator's distribution $p_g$).

Generator Loss Function (standard loss):

$$L_G = -\mathbb{E}_{z \sim p_z}[\log(D(G(z)))]$$

Figure 2: Generator Loss Function

Generator Gradient:

$\frac{1}{D(G(z))}$: The generator receives stronger feedback, or in other terms, larger gradients when $D(G(z))$ is small, i.e. when the discriminator believes the sample is fake).

$$\nabla_{\theta_G} L_G = -\mathbb{E}_{z \sim p_z} \left[ \frac{1}{D(G(z))} \nabla_{\theta_G} D(G(z)) \right].$$

Figure 3: Generator Gradient

$\nabla_{\theta_g} D(G(z))$: The chain rule is applied since $D(G(z))$ relies on the generator's parameters $\theta_G$ through the generated sample $G(z)$.

The discriminator learns to maximize the first (correctly classifying real data) and second term (correctly classifying fake data) in the loss function.

Discriminator Objective Function:

$$\mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{x \sim p_g}[\log(1 - D(x))].$$

Figure 4: Discriminator Objective Function

$E_{x \sim p_d}$: This term evaluates how well the discriminator assigns high confidence $D(x) \to 1$ to real data samples $x \sim p_d$, from the data distribution.

Maximizing this term encourages the discriminator to correctly classify true samples as real.

$\mathbb{E}_{x \sim p_g}[\log(1 - D(x))]$: This term evaluates how well the discriminator assigns low confidence $D(x) \to 0$ to fake samples $x \sim p_g$, made by the generator.

Maximizing this term encourages the discriminator to correctly classify generated samples as fake.

Discriminator Loss Function:

$$L_D = -\mathbb{E}_{x \sim p_d}[\log D(x)] - \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

Figure 5: Discriminator Loss Function

$D(x)$: The discriminator's output for a real sample x, the probability x is real.

$D(G(z))$: The discriminator's output for a fake sample $G(z)$, generated from the generator using noise $z$, sample from the prior distribution $p_z$.

$\mathbb{E}_{x\sim p_d}[\log D(x)]$: The expected value of the log probability of classifying true samples correctly.

Discriminator Gradient:

$\mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$: The expected value of the log probability of classifying fake samples correctly.

$$\nabla_{\theta_D} L_D = -\mathbb{E}_{x\sim p_d}\left[\frac{\nabla_{\theta_D} D(x)}{D(x)}\right] - \mathbb{E}_{z\sim p_z}\left[\frac{\nabla_{\theta_D} D(G(z))}{1 - D(G(z))}\right]$$

Figure 6: Discriminator Gradient

The gradient updates $\theta_D$ to:

Increase $D(x)$ (output for true samples) so that $logD(x)$ becomes larger.

Also to decrease $D(G(z))$ (output for false samples) so $log(1 - D(G(z))$ becomes larger.

# 4 Results

## 4.1 Model Architecture

We built our system on top of the Stable Diffusion model, consisting of a frozen Variational Autoencorder (VAE) and a frozen CLIP-based text encoder, all combined with a UNet-based diffusion model. The VAE encodes images into a latent space, and the CLIP text encoder transforms text-based prompts into latent text embeddings. The UNet learns to denoise random latent vectors conditioned on the text embeddings. We 'fine-tune' the Stable Diffusion v1.4 weights and only update the UNet parameters (weights and biases) while keeping the VAE and CLIP text encoder froze. This allows us to take advantage of strong priors learned by the Stable Diffusion model and let the UNet to adapt towards our new target domain (celebrity faces).

## 4.2 Fine-Tuning Procedure

The training script, `train.py`, is where our UNet fine-tuning occurs. We load in the pre-trained stable diffusion components (`CompVis/stable-diffusion-v1-4`) and freezes the VAE and text encoder to keep their pre-trained representations. We leave only the UNet parameters trainable. The training dataset is CelebA, which contains about $\tilde{2}$00,000 images of celebrity faces. We used a generic prompt ("A high-resolution photo of a face") to condition the generation process.

The training loop follows a standard diffusion training process:

1. **Prepare the Data**:
   Each image is encoded into a latent space using the frozen VAE. The text prompt is tokenized and encoded using the frozen CLIP text encoder.

2. **Denoising Objective**:
   After data preparation, we add random noise to the latent representation and use the UNet to predict the noise. We then calculate a mean-squared error (MSE) loss between the predicted noise and the actual noise driving the weight updating.

At the end of training, we save the updated UNet weights, which creates a new version of Stable Diffusion that is fine-tuned in generating high quality images of faces

## 4.3 Generating Images

The `generate.py` script loads in our new fine-tuned UNet, in addition to the original VAE and text encoder, and generates a large set of images according to a given prompt.

The process entails:

1. **Loading the Fine-Tuned UNet**:
   The script loads in our final model generated from the fine tuning script.

2. **Stable Diffusion Inference**:
   We use the Stable Diffusion pipeline to sample images conditioned on the same prompt (or a new prompt if you so choose) and produced a set of image (2,900 for our testing purposes). Each generation is guided by our textual prompt and generated in 50 diffusion steps with a guidance scale of 7.5 to ensure the final image is aligned with the prompt and still maintain some natural diversity.

3. **Saving the Results**:
   The generated images are then stored locally for later evaluation and analysis.

## 4.4 FID Evaluation

To quantitatively measure the quality of the generated images and how well they resemble the real faces from the CelebA dataset, we computed the Frechet Inception Distance (FID). The FID compares the distribution of generated images against the distribution of real images.

After generating 2,900 images with our fine tuned model and comparing them to the original CelebA dataset, we obtained an FID score of **59.6**. A lower FID score indicates better fidelity and diversity, our obtained FID indicates room for improvement.

A score of $\tilde{5}9$ indicates that our model produces images similar to those found in the CelebA dataset, however it also tells us that there is a noticeable gap in the alignment of quality between our generated faces and the CelebA dataset.

A likely contributing factor is that our generated images sometimes slightly tilted or cropped, which differs from the consistently framed faces in the CelebA dataset. The FID score likely suffered from such inconsistencies. Subtle misalignmnets can cause the differences in the expected facial structure and thus negatively impacting the Inception-based FID score.

### 4.5 Potential Remedies

1. **Cropping and Alignment Adjustments**:
   Ensure that generated images have a similar orientation and framing like those of the CelebA dataset. This could be accomplished some kind of post-processing technique like face alignment and incorporate a training-time penalty for off-angle faces. This could reduce discrepancies between distributions.

2. **Prompt Adjustment**:
   We could adjust the prompt to emphasize a frontal and properly aligned to generate more consistently framed images. For example, the prompt could be adjusted to "A front facing, well aligned, high-resolution photo of a face."

## 5 Conclusion

The results of fine-tuning show that although that we were able to adapt the Stable Diffusion model to generate realistic headshots from text text prompts, the FID score of $\tilde{5}9$ indicates that there is still some progress left to be desired to close the gap between the generated data and real data distributions. The FID score indicatest that there are issues witht the subtle misalignments such as tilted and cropped images. Despite this, the model shows that it can produce a diverse set of facial images, highlighting the effectiveness of starting from a large, pretrained model and fine tuning it towards a particular domain of images.

This project shows that adjustments to a large pretrained model can generate impressive domain specific results. As larger diffusion models become more accessible, this project shows how users of these models can fine tune them on their own datasets to suit their own needs rather then training an entire model from scratch. This is good because it lowers the barriers to entry (such as cost)

for being able to take advantage of large diffusion models. This can lead to many positive outcomes, such as accelerating research.

Future improvements include refining prompts for better framing, use some kind of data augmentation or face alignment to standardize outputs, and adjustin training parameters. As the method of 'fine-tuning' existing models progresses to create specialized, fine-tuned versions of existing models becomes more popular and advanced, domain-optimized generative capabilties can become even more accessible to the public.

# 6 References

# References


[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, arXiv preprint arXiv:1406.2661, 2014. Retrieved from `http://www.github.com/goodfeli/adversarial`.

[2] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, *Diffusion models: A comprehensive survey of methods and applications*, arXiv preprint arXiv:2209.00796, 2023. Retrieved from `https://github.com/YangLing0818/Diffusion-Models-Papers-Survey-Taxonomy`.


# 7 Links

1. Github Repository with Scripts You Need
2. OneDrive link with samples from the real and generated dataset, along with the full original dataset