Generating Synthetic Hand-Sketched Images Using VAE, DCGAN, and WGAN

CSE 429: Computer Vision and Pattern Recognition

Department of Computer Science and Engineering

Date: May 20, 2025

Group Members:

Hesham Ahmed Ebaid 120210064

Ahmed Medhat Loutfy 120210063

Yehia Ali Othman 120210302

Abstract

Generating hand-sketched facial images is a challenging problem with applications in artistic creation and forensics. We implement and compare three deep generative models—Variational Autoencoder (VAE), Deep Convolutional GAN (DCGAN), and Wasserstein GAN (WGAN)—trained on the CUHK Face Sketch dataset. Our results indicate that WGAN achieves the best quality sketches, and we provide a Flask-based web application for interactive sketch generation.

Teaser Figure

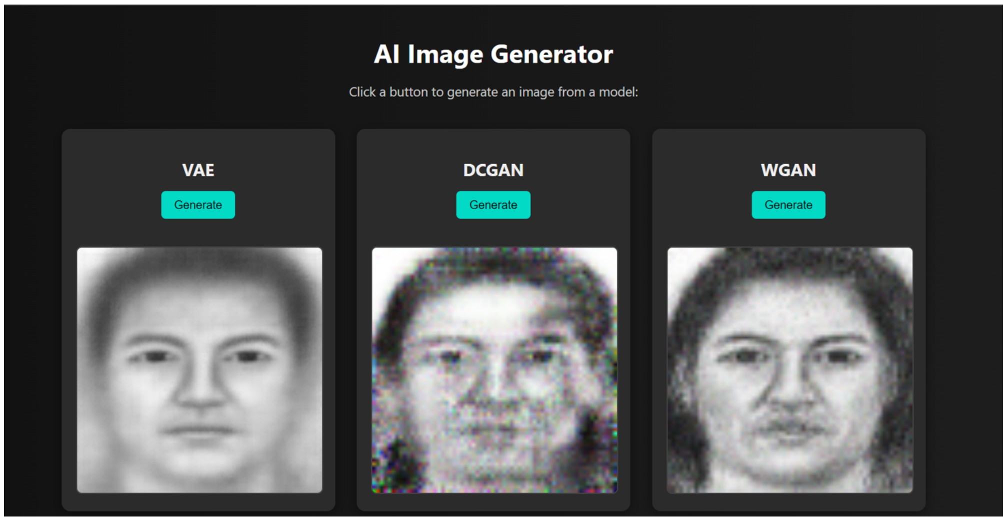


Figure 1: Comparison of synthetic hand-sketched facial images generated by VAE, DCGAN, and WGAN.

1 Introduction

Hand-drawn sketches are widely used in art, forensics, and human-computer interaction. Automating sketch generation can accelerate creative workflows and aid in facial recognition and law enforcement. Unlike photographic images, sketches require capturing stylistic abstractions, posing unique challenges for generative models.

Traditional generative models often focus on photorealistic image synthesis. However, sketch generation demands learning style and texture patterns that differ significantly from natural images. This project compares three generative models—VAE, DCGAN, and WGAN—for sketch synthesis and integrates them into a user-friendly Flask web application for interactive sketch generation.

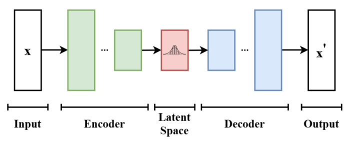
2 Approach

2.1 Dataset and Preprocessing

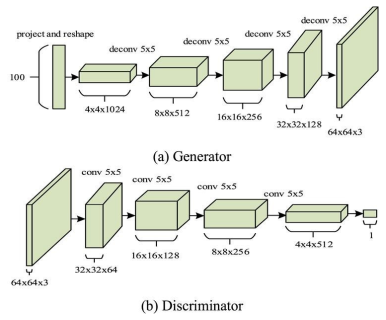
We used the CUHK Face Sketch (CUFS) dataset, which contains paired face photos and sketches. Images were resized to 128×128 pixels, converted to grayscale, and normalized to a range suitable for the models.

2.2 Model Architectures

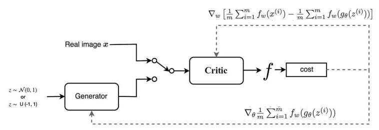
- **Variational Autoencoder (VAE):** An encoder-decoder structure trained with reconstruction loss to generate sketch-like images from latent vectors.
- **Deep Convolutional GAN (DCGAN):** A convolutional GAN trained adversarially to produce realistic sketches.
- **Wasserstein GAN (WGAN):** An improved GAN variant using Wasserstein loss and gradient penalty to improve training stability and output quality.



(a) VAE Architecture



(b) DCGAN Architecture



(c) WGAN Architecture

Figure 2: Architectural diagrams of the generative models used in this study.

2.3 Training Details

Each model was trained independently for 100 epochs with a batch size of 64 using the Adam optimizer. Hyperparameters such as learning rates were tuned per model to optimize performance. We encountered challenges such as GAN training instability and mode collapse, which were addressed by careful tuning and implementing gradient penalty in WGAN.

2.4 Web Application

The Flask backend loads the trained models and generates sketch images based on user input selecting one of the models. The frontend uses simple HTML and CSS with no client-side scripting, ensuring fast responses and easy deployment.

3 Experiments and Results

3.1 Experimental Setup

- **Dataset size:** 606 images split 80% training, 20% testing.
- **Image resolution:** 256×256 grayscale sketches.
- **Metrics:** Fréchet Inception Distance (FID) and visual inspection.

3.2 Quantitative Results

Table 1: Model performance comparison

Model	FID Score (Lower Better)	Training Stability	Visual Quality
VAE	200.128	High	Blurry, less detailed
DCGAN	191.055	Moderate	Sharper but unstable
WGAN	92.940	High	Sharp, coherent sketches

3.3 Understanding the FID Metric

To quantitatively evaluate the similarity between real and generated sketch images, we use the Fréchet Inception Distance (FID). This metric measures the distance between the distributions of feature representations of real and generated images, extracted using a pretrained InceptionV3 model.

Step-by-Step FID Calculation

1. **Feature Extraction:** Real and generated images are passed through InceptionV3, and feature vectors (2048-dimensional) are extracted from the penultimate layer.
2. **Compute Statistics:** For each set of features, compute the mean vector μ and the covariance matrix Σ.
3. **Calculate Fréchet Distance:**

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

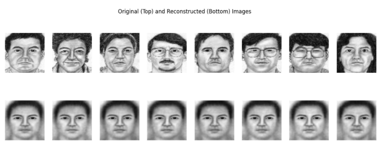
   where μ_r, Σ_r and μ_g, Σ_g correspond to the mean and covariance of real and generated image features, respectively.

4. **Interpretation:** Lower FID values indicate that the distribution of generated features is closer to real ones, reflecting better image quality and diversity.

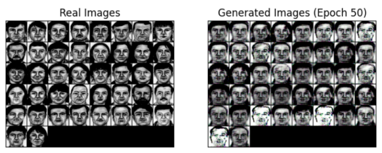
We implemented this using PyTorch and NumPy. All images were preprocessed (resized and normalized) to match the InceptionV3 input requirements.

3.4 Qualitative Results

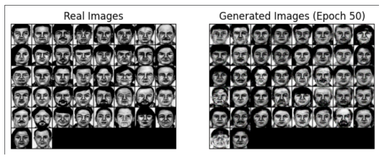
Successes include realistic facial sketches by WGAN and DCGAN, while VAE outputs were often blurry. DCGAN sometimes suffered from mode collapse.



(a) VAE Output



(b) DCGAN Output



(c) WGAN Output

Figure 3: Examples of generated sketches from each model.

4 Conclusion and Future Work

This project successfully demonstrated the generation of hand-sketched facial images using VAE, DCGAN, and WGAN models. WGAN achieved the highest quality and stability. Our Flask web app allows users to interactively generate sketches and compare models.

Future improvements include conditional sketch generation based on input photos, interactive refinement tools, and integrating more advanced architectures such as StyleGAN to improve detail and variety.

References

1. Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.

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

3. Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.
4. Wang, X., et al. (2012). Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
5. Flask Documentation.
6. PyTorch Documentation.