Project Report - ECE 176

Juqy Chen
Department of Electrical and Computer Engineering
A17796879

Yilei Wang
Department of Mathematics
A18307565

Abstract

This project compares Deep Convolutional GAN (DCGAN) and Wasserstein GAN with Gradient Penalty (WGAN-GP) for the task of generating realistic face images. We implement both methods from scratch (**see GitHub repository here**) and experimentally evaluate their training stability, loss dynamics, and image quality. Results indicate that while DCGAN can learn to generate plausible faces, it often suffers from training instability and mode collapse. In contrast, WGAN-GP demonstrates smoother convergence and produces sharper, more diverse face images. This report outlines our objectives, methodology, experiments, and findings.

1 Introduction

Generative Adversarial Networks (GANs) have become a cornerstone for image synthesis tasks. While it is widely known that basic GANs can perform well for simple tasks like digit generation, they often struggle to capture the intricate details needed for realistic face synthesis.In our project, we started with DCGAN (Radford et al.) to leverage convolutional architectures for better feature extraction. Although DCGAN significantly improved feature learning, it introduced a new set of challenges. Notably, the training process became highly unstable; the loss would oscillate continuously without effective convergence. Moreover, we found that after approximately 30,000 iterations, the DCGAN seemed to reach a plateau where it could no longer learn or refine new features. Motivated by these challenges, we explored the use of Wasserstein GAN with Gradient Penalty (WGAN-GP). The results were markedly better: WGAN-GP not only enabled the model to capture and learn a richer set of features, but it also maintained steady convergence even beyond the 30,000-iteration mark. This indicates that WGAN-GP is better suited for tasks that require the synthesis of complex and detailed images, such as human faces.

2 Related Work

Our work is primarily inspired by several key papers:

- **Wasserstein GAN (WGAN)** by Arjovsky et al. (2017), which introduced the concept of using the Wasserstein distance for GAN training.
- **Improved Training of Wasserstein GANs (WGAN-GP)** by Gulrajani et al. (2017), which proposed the gradient penalty to enforce the 1-Lipschitz constraint.
- **DCGAN** by Radford et al. (2015), which provided a solid baseline for GAN architectures.

These works collectively contribute to our understanding of GAN training stability and inform our design decisions.

3 Method

Our method builds upon the WGAN-GP framework. The key components are as follows:

3.1 Wasserstein Loss with Gradient Penalty

The critic (discriminator) is designed to differentiate between real and generated images. Its original objective is to maximize the difference:

$$\max_{D} \quad \mathbb{E}_{x \sim p_{data}}[D(x)] - \mathbb{E}_{z \sim p_z}[D(G(z))],$$

which encourages the critic to assign high scores to real images and low scores to generated images.

To enforce the necessary 1-Lipschitz condition on D, we add a gradient penalty term:

$$\lambda_{GP}\, \mathbb{E}_{\hat{x}} \left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2 \right],$$

where $\hat{x}$ is an interpolation between real and generated images and λ_{GP} is the gradient penalty coefficient.

Thus, the overall critic loss (which we minimize) becomes:

$$L_D = -\mathbb{E}[D(x)] + \mathbb{E}[D(G(z))] + \lambda_{GP}\, \mathbb{E}_{\hat{x}} \left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2 \right].$$

Similarly, the generator is trained to maximize the critic's score on the generated images, which is equivalent to minimizing:

$$L_G = -\mathbb{E}[D(G(z))].$$

Detailed Explanation:

- **Critic Objective:** The critic aims to maximize $\mathbb{E}[D(x)] - \mathbb{E}[D(G(z))]$. In our implementation, we minimize L_D, which, due to the negative signs, corresponds to maximizing the original objective. The gradient penalty term ensures that D satisfies the 1-Lipschitz constraint by penalizing deviations of the gradient norm from 1.
- **Generator Objective:** The generator minimizes $L_G = -\mathbb{E}[D(G(z))]$. Minimizing L_G means that the generator is encouraged to produce images that receive high scores from the critic. If the critic rates the generated images positively, L_G becomes negative, which indicates improved performance by the generator.
- **Interpretation of Negative Loss Values:**
  - A **negative generator loss** indicates that the average critic score for the generated images is positive. This is desirable, as it shows the generator is producing images that are closer to the real data distribution.
  - A **negative critic loss** (in the minimized formulation) suggests that the critic is effectively distinguishing between real and generated images while adhering to the Lipschitz constraint.

3.2 Network Architecture

We design the generator and discriminator (critic) with modular code structure:

- **Generator:** Uses a series of transposed convolutions (ConvTranspose2d) with Batch Normalization and ReLU activations, ending with a Tanh activation to generate images in the range [-1,1].
- **Discriminator:** Consists of convolutional layers with Instance Normalization (or Layer Normalization, as recommended in WGAN-GP) and LeakyReLU activations. It outputs a scalar score for each image.

Below, we detail the architectures for both the Generator and the Discriminator used in our WGAN-GP model.

Table 1: Generator Architecture

Layer	Operation	Input Shape	Output Shape	Activation / Normalization
1	FC & Reshape	$z \in \mathbb{R}^{100}$	$4 \times 4 \times 512$	–
2	Deconv (4,2,1)	$4 \times 4 \times 512$	$8 \times 8 \times 256$	Batch Norm + ReLU
3	Deconv (4,2,1)	$8 \times 8 \times 256$	$16 \times 16 \times 128$	Batch Norm + ReLU
4	Deconv (4,2,1)	$16 \times 16 \times 128$	$32 \times 32 \times 64$	Batch Norm + ReLU
5	Deconv (4,2,1)	$32 \times 32 \times 64$	$64 \times 64 \times 3$	Tanh

Table 2: Discriminator (Critic) Architecture

Layer	Operation	Input Shape	Output Shape	Activation / Normalization
1	Conv (4,2,1)	$64 \times 64 \times 3$	$32 \times 32 \times 64$	LeakyReLU
2	Conv (4,2,1)	$32 \times 32 \times 64$	$16 \times 16 \times 128$	Instance Norm + LeakyReLU
3	Conv (4,2,1)	$16 \times 16 \times 128$	$8 \times 8 \times 256$	Instance Norm + LeakyReLU
4	Conv (4,2,1)	$8 \times 8 \times 256$	$4 \times 4 \times 512$	Instance Norm + LeakyReLU
5	Fully Connected	Flattened $4 \times 4 \times 512$	Scalar Score	–

3.3 Hyperparameter Optimization

We integrate Optuna to automatically search for the optimal values of:

- Learning rates for both the generator and critic.
- The number of critic updates per generator update (n_critic).
- The gradient penalty coefficient (λ_{GP}).

4 Experiments

In this section, we describe the datasets, preprocessing, training details, results, and an ablation study to evaluate the effectiveness of key components in our method.

4.1 Dataset and Preprocessing

We use the CelebA dataset, which contains over 200,000 face images. This dataset is widely used for face synthesis tasks due to its diversity in facial attributes.

- **Data Format:** Images are stored in JPEG format.
- **Preprocessing:**
  - All images are resized to 64×64 pixels.
  - A center-crop is applied to ensure proper face alignment.
  - The images are normalized to the range [-1, 1] to match the generator output using the Tanh activation.

4.2 Training Details

We implement and train both DCGAN and WGAN-GP models using PyTorch. The following settings were used for the WGAN-GP model:

- **Batch Size:** 64.
- **Optimizers:** Adam with parameters $\beta_1 = 0$ and $\beta_2 = 0.9$.
- **Learning Rates:** Specific values are tuned via hyperparameter optimization
- **n_critic:** The critic is updated 7 times for every generator update.
- **Gradient Penalty:** The coefficient λ_{GP} is set to 18.
- **epoch:** Models are trained for 25 epochs in total.

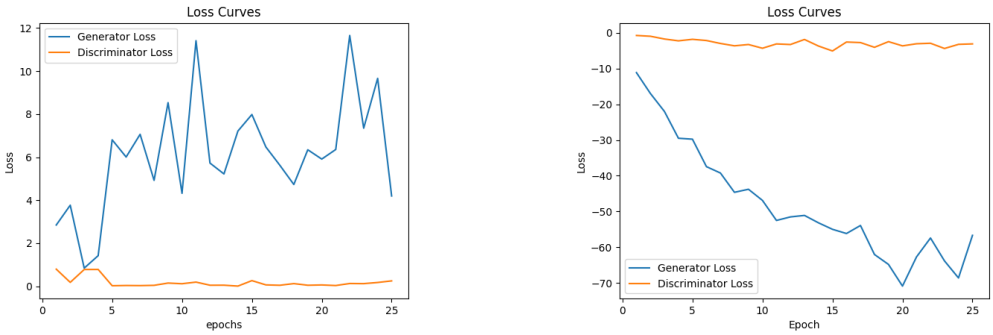
4.3 Results

We evaluate our models using both qualitative and quantitative metrics:

- **Visual Inspection:** Sample images generated at various training iterations are shown in Figure 1.
- **Loss Dynamics:** The evolution of generator and critic losses of our models in Figure 2.

(a) DCGAN at 25 Epochs (b) WGAN-GP at 25 Epochs

Figure 1: Comparison of generated images from DCGAN and WGAN-GP after 25 epochs of training.



(a) Loss curve for DCGAN (b) Loss curve for WPGAN-GP

Figure 2: Comparison of generated images with DCGAN and WPGAN-GPs.

4.4 Ablation Study

To understand the contribution of each component in our method, we conducted an ablation study by varying key hyperparameters:

- **n_critic:** We experimented with values 10, 15, and 20. We observed that a higher n_critic leads to more stable training, for our model we use 18 as critic.
- **Gradient Penalty (λ_{GP}):** We tested values of 1, 5, and 10. Our results indicate that $\lambda_{GP} = 8$ provides the best balance between training stability and performance.
- **Learning Rate:** Various learning rates were tested for both the generator and critic. The optimal values were selected based on the quality of generated images and loss convergence.

Future work will further explore additional metrics and training configurations to enhance model performance.

5 Supplementary Material

A video presentation summarizing the project motivation, approach, and key results will be provided.

References

[1] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

[2] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved Training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*.

[3] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*.