Université Paris Cité

Deep Learning

Comparative Study of WGAN and DDPM for Generative Image Restoration Tasks

Keïn Sam
Hazim Baroudi

May 2025

Led by
Ilaria Giulini and Arthur Stephanovitch

Contents

Chapter 1

Introduction

Generative modeling has emerged as a central paradigm in modern deep learning, enabling machines to synthesize high-quality data samples from learned distributions. Among the most prominent frameworks are Generative Adversarial Networks (GANs) and Score-Based Generative Models (SGMs), which represent two fundamentally different approaches to learning generative distributions.

Models studied: This project focuses on a comparative study of two state-of-the-art models from each family: the Wasserstein Generative Adversarial Network (WGAN) and the Denoising Diffusion Probabilistic Model (DDPM). While WGANs leverage an adversarial objective to push the generated distribution closer to the target one via a critic function, DDPMs rely on a gradual denoising process to implicitly learn the score function of the data distribution.

Application tasks: We evaluate and compare these models on three image restoration tasks that are inpainting, outpainting, and super-resolution. Each task presents unique challenges in learning a conditional distribution $p(x \mid y)$, where y is an observed degraded or partial version of the target image x.

Project goals:

- Understand the mathematical foundations of WGAN and DDPM, from first principles to modern conditional generation techniques.
- Formulate and implement each of the three generation tasks in a controlled setting using both models.
- Perform a quantitative and qualitative comparison between the two approaches across tasks, highlighting their respective strengths and limitations.

The code is available here : https://github.com/keinsam/generative-models-comparison.

Chapter 2

Presentation of the Models

In this section, we present the mathematical foundations of the generative models studied in this project. We begin with Generative Adversarial Networks (GANs) and their improvement into Wasserstein Generative Adversarial Networks (WGANs). We then cover Score-Based Generative Models (SGMs), leading into Denoising Diffusion Probabilistic Models (DDPMs).

Let $x \in \mathbb{R}^d$ denote a data point drawn from an unknown data distribution p_{data}. Generative models aim to learn a model distribution p_G that approximates p_{data}.

2.1 From GAN to WGAN

2.1.1 Generative Adversarial Networks (GAN)

Let $z \in \mathbb{R}^k$ be a latent variable sampled from a prior distribution p_z, typically $\mathcal{N}(0, I_k)$. The generator $G : \mathbb{R}^k \to \mathbb{R}^d$ maps z to the data space. The discriminator $D : \mathbb{R}^d \to [0, 1]$ estimates the probability that a sample is real.

The original minimax optimization problem is

$$\min_G \max_D -\mathcal{L}_D$$

where $\mathcal{L}_D = -\left(\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]\right)$ is the discriminator loss to minimize (or maximize its negative). The generator loss to minimize is $\mathcal{L}_G = -\mathbb{E}_{z\sim p_z}[\log D(G(z))]$.

This minimax corresponds to minimizing the Jensen-Shannon divergence $\text{JS}(p_{\text{data}} \| p_G)$, where p_G is the distribution induced by $G(z)$. However, when $\text{supp}(p_{\text{data}}) \cap \text{supp}(p_G) = \emptyset$, gradients vanish, leading to unstable training.

2.1.2 Wasserstein Generative Adversarial Networks (WGAN)

To address this, WGAN replaces the JS divergence with the Wasserstein-1 distance. By the Kantorovich-Rubinstein duality:

$$W(p_{\text{data}}, p_G) = \sup_{\|C\|_L \leq 1} \mathbb{E}_{x\sim p_{\text{data}}}[C(x)] - \mathbb{E}_{z\sim p_z}[C(G(z))]$$

where $C : \mathbb{R}^d \to \mathbb{R}$ is a 1-Lipschitz function, often called the "critic" (instead of discriminator). In practice, this 1-Lipschitz condition is enforced using weight clipping or gradient penalty (WGAN-GP).

The critic loss is given by:

$$\mathcal{L}_C = -(\mathbb{E}_{x\sim p_{\text{data}}}[C(x)] - \mathbb{E}_{z\sim p_z}[C(G(z))])$$

The generator loss is:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z}[C(G(z))]$$

Logically, the original minimax optimization problem becomes $\min_G \max_C -\mathcal{L}_C$.

2.2 From SGM to DDPM

2.2.1 Score-Based Generative Models (SGM)

The score function of a distribution $p(x)$ is $s^*(x) := \nabla_x \log p(x)$.

For $\sigma > 0$, perturb the data as $\tilde{x} = x + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, I_d)$. A neural network $s_\theta(x, \sigma)$ is trained to approximate the score, with a denoising objective:

$$\mathcal{L}_{\text{SGM}}(\theta) = \mathbb{E}_{i\sim\mathcal{U},x\sim p_{\text{data}},\epsilon\sim\mathcal{N}(0,I_d)}\left[\lambda(\sigma_i)\left\|s_\theta(x+\sigma_i\epsilon,\sigma_i)+\frac{\epsilon}{\sigma_i}\right\|^2\right]$$

Sampling uses Langevin dynamics:

$$x_{t+1} = x_t + \frac{\eta}{2}s_\theta(x_t, \sigma_t) + \sqrt{\eta}\cdot\epsilon_t$$

SGMs simulate a stochastic differential equation (SDE) that transforms noise into data, but their continuous solution can be unstable and difficult to optimize.

2.2.2 Denoising Diffusion Probabilistic Models (DDPM)

DDPMs define a forward diffusion process $\{x_t\}_{t=0}^T$ with Gaussian noise:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I_d)$$

with $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$, so:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I_d)$$

The generative model learns to reverse this process, predicting noise ϵ and using:

$$\mathcal{L}_{\text{DDPM}}(\theta) = \mathbb{E}_{t\sim[1,T],x_0\sim p_{\text{data}},\epsilon\sim\mathcal{N}(0,I_d)}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right]$$

Sampling iteratively:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right) + \sqrt{\beta_t}\cdot z$$

where $z \sim \mathcal{N}(0, I_d)$.

Thus, DDPMs are discrete-time approximations of SGMs, learning the noise function $\epsilon_\theta(x_t, t)$ by estimating the time-dependent score $\nabla_{x_t} \log p_t(x_t)$ and where the reverse process is derived using the Euler-Maruyama scheme.

Chapter 3

Presentation of the Image Restoration Tasks

This chapter presents the mathematical formulation of the tasks used to compare WGAN and DDPM: inpainting, outpainting, and super-resolution. For each, we describe how WGANs and DDPMs are adapted to conditional generation.

Let $x \in \mathbb{R}^{H \times W \times C}$ be a full image, and y the conditional input (masked version, crop, or low-resolution). The goal is to approximate $p_\theta(x \mid y)$.

3.1 Image Inpainting

Problem: We observe a corrupted image $y = M \odot x$, where $M \in \{0,1\}^{H \times W}$ is a binary mask. The task is to reconstruct x from y.

WGAN-based Inpainting:

The generator $G(z, y)$ is conditioned on the masked input y, and the critic $C(x, y)$ evaluates whether an image x is real or generated given the input y. The Wasserstein loss is:

$$\mathcal{L}_{\text{C-inpaint}} = -(\mathbb{E}[C(x, y)] - \mathbb{E}[C(G(z, y), y)])$$

$$\mathcal{L}_{\text{G-inpaint}} = -\mathbb{E}[C(G(z, y), y)]$$

A reconstruction term is added to ensure that the model focuses on the masked part of the image:

$$\mathcal{L}_{\text{recon}} = \|(M \odot x + (1 - M) \odot G(z, y)) - x\|_1$$

Final generator loss:

$$\mathcal{L}_{\text{G-inpaint}} = -\mathbb{E}[C(G(z, y), y)] + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}}$$

where λ_{recon} is a new hyperparameter controlling how much emphasis is placed on the reconstruction term.

DDPM-based Inpainting:

The noise predictor $\epsilon_\theta(x_t, t, y)$ is conditioned on y, typically by masking or concatenating it. The training loss is the sum:

$$\mathcal{L}_{\text{DDPM-inpaint}} = \mathbb{E}_{x,\epsilon,t}\left[\|(1-M)\odot(\epsilon-\epsilon_\theta(x_t,t,y))\|^2\right]$$

Only the unobserved regions are predicted.

3.2 Image Outpainting

Problem: Given a central crop $y = \mathcal{R}(x)$, predict the full image x. We can see $\mathcal{R}(x) = M \odot x$ as a masked image, similar to the inpainting task.

WGAN-based Outpainting:

The generator $G(z, y)$ generates full images conditioned on y, and the discriminator evaluates realism. The losses are: $\mathcal{L}_{\text{C-outpaint}} = \mathcal{L}_{\text{C-inpaint}}$ and $\mathcal{L}_{\text{G-outpaint}} = -\mathbb{E}[C(G(z,y),y)] + \lambda_{\text{recon}}\|\mathcal{R}(G(z,y)) - y\|_1$.

DDPM-based Outpainting:

The crop y is encoded and injected into x_t at each timestep, either via concatenation or masking. The training loss is $\text{L}_{\text{DDPM-outpaint}} = \mathbb{E}\left[\|\mathcal{R}(\epsilon-\epsilon_\theta(x_t,t,y))\|^2\right]$.

3.3 Image Super-Resolution

Problem: We aim to recover a high-resolution image $x \in \mathbb{R}^{H\times W\times C}$ from our low-resolution input $y \in \mathbb{R}^{\frac{H}{s}\times\frac{W}{s}\times C}$, where s is the downscaling factor.

WGAN-based Super-Resolution:

The generator $G(z, y)$ receives a low-resolution image y upsampled (via bilinear interpolation) and generates a high-resolution image. The critic $C(x, y)$ compares real high-resolution images with generated ones, both conditioned on y. The loss is:

$$\mathcal{L}_{\text{C-sr}} = -(\mathbb{E}[C(x,y)] - \mathbb{E}[C(G(z,y),y)])$$

$$\mathcal{L}_{\text{G-sr}} = -\mathbb{E}[C(G(z,y),y)] + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}}$$

with $\mathcal{L}_{\text{recon}} = \|x - G(z,y)\|_1$.

This reconstruction term helps preserve fine details and stabilizes training.

DDPM-based Super-Resolution:

In the DDPM setting, the noise predictor $\epsilon_\theta(x_t, t, y)$ is conditioned on the low-resolution input y, typically upsampled and concatenated with the noisy sample x_t. The loss is computed over the entire image:

$$\mathcal{L}_{\text{DDPM-sr}} = \mathbb{E}_{x,\epsilon,t}\left[\|\epsilon-\epsilon_\theta(x_t,t,y)\|^2\right]$$

This setup allows the model to learn how to denoise while respecting the structure provided by the low-resolution condition.

Chapter 4

Implementation of the Models and Datasets

Now that we have established the mathematical foundations of our models and tasks, we proceed to the implementation details. To enable consistent comparison across tasks, we first implemented a basic (non-conditional) image generation pipeline using standard datasets and models. Each task-specific model and dataset inherits from this base implementation. Consequently, we place particular emphasis on the design of our base classes, implemented in PyTorch.

4.1 Basic Implementations

We define core parent classes that serve as the foundation for all task-specific datasets and models. This modular structure promotes code reuse and ensures fair comparisons across tasks. We also maintain architectural similarity and parameter count parity across models.

4.1.1 BaseCIFAR10

We used a subset of the CIFAR10 dataset for our experiments and called it BaseCIFAR10. It is made up of 10 000 RGB images of 32x32 pixels.

4.1.2 BaseWGAN

BaseCritic Architecture

The BaseCritic is implemented as a convolutional neural network and is composed of the following:

- **Input:** An image tensor, either x or $G(z)$.
- **Hidden Layers:** Three downsampling blocks, each composed of a `Conv2d` layer, `BatchNorm2d` normalization and `LeakyReLU` activations.
- **Output:** A `Conv2d` layer mapping to a single scalar.

Weight clipping is applied during training to enforce the 1-Lipschitz constraint.

BaseGenerator Architecture

The BaseGenerator tries to mirror the critic's architecture and is composed of:

- **Input:** A latent tensor z.
- **Hidden Layers:** Three upsampling blocks, each composed of a `ConvTranspose2d` layer, `BatchNorm2d` normalization and `LeakyReLU` activations.
- **Output:** A final `Conv2d` layer with a `Tanh` activation to scale pixel values to $[-1, 1]$.

4.1.3 BaseDDPM

Noise Schedule

We implemented a linear noise schedule for the β_t coefficients between fixed bounds $\beta_{\min}$ and $\beta_{\max}$, precomputing all necessary coefficients (α_t, $\bar{\alpha}_t$) for efficient sampling and training.

BaseEpsilonUNet Architecture for ϵ_θ

The denoising model is a U-Net with skip connections and time conditioning (but no self-attention):

- **Input:** A noisy image x_t and timestep t tensors.
- **Time Embedding:** t is passed through a sinusoidal positional encoding and an MLP.
- **Encoder:** Three downsampling blocks, each composed of `MaxPool2d`, a `Conv2d` layer, `BatchNorm2d` normalization and `LeakyReLU` activations.
- **Bottleneck:** A a `Conv2d` layer that combines spatial features with the time embedding.
- **Decoder:** Three upsampling blocks with skip connections from the encoder, each block is composed of `Upsample`, a `ConvTranspose2d` layer, `BatchNorm2d` normalization and `LeakyReLU` activations.
- **Output:** A `Conv2d` layer mapping to a noise tensor of the same shape as x_t.

In the forward process, the time embedding is summed to the bottleneck's input.

Sampling Procedure

We use a sampling loop using the precomputed noise schedule. Starting from $x_T \sim \mathcal{N}(0, I)$, the model is called repeatedly to estimate $\epsilon_\theta(x_t, t)$ and compute x_{t-1} using the reverse diffusion formula.

4.2 Task-Specific Implementations

4.2.1 Task-Specific Datasets

As explained previously, each task-specific dataset (`InpaintingCIFAR10`, `OutpaintingCIFAR10`, and `SuperResolutionCIFAR10`) inherits from the `BaseCIFAR10` class. These datasets provide both the ground truth image x and the conditioning variable y, which corresponds to masked images or low-resolution versions, depending on the task. They are designed to deliver the input format required for solving each task, as detailed in the previous chapter.

4.2.2 Task-Specific Models

Similarly, task-specific models (`<Task_Name><Model_Name>`) inherit from their corresponding `Base<Model_Name>` class. Most of the necessary modifications are implemented in the forward pass to account for the conditioning mechanism. Training is then carried out using the appropriate loss functions introduced in the previous chapter.

Chapter 5

Experimental Results

We now present the experimental results comparing WGANs and DDPMs across three conditional image restoration tasks. For each task, we provide both qualitative results (sample generation from each model) and quantitative metrics: Fréchet Inception Distance (FID), Structural Similarity Index (SSIM), and L1 reconstruction error.

For all models, training is performed using the Adam optimizer for 200 epochs. However, the learning rate and other hyperparameters were selected independently to ensure stable training and a fair comparison. We began with the hyperparameters from the original papers and then tuned them for our models.

5.1 InpaintingWGAN vs. InpaintingDDPM

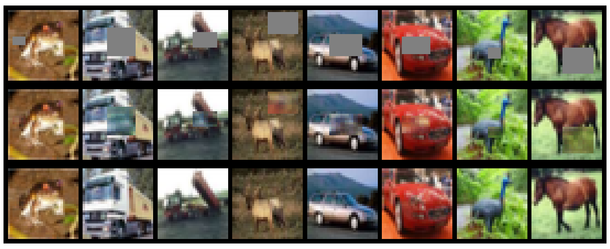
Qualitative interpretation: As shown in Figure 5.1, the inpainted regions produced by the InpaintingDDPM tend to be noisy and lack smoothness. In contrast, the InpaintingWGAN generates smoother outputs, but a noticeable contrast often remains between the masked and unmasked regions of the image.

Quantitative results: Table 5.1 indicates that both models achieve similar SSIM and L1 scores, suggesting comparable pixel-wise and structural reconstruction quality. However, the InpaintingDDPM outperforms in terms of FID, which can be attributed to the model's ability to better capture the data distribution.

5.2 OutpaintingWGAN vs. OutpaintingDDPM

Qualitative interpretation: Regarding the outpainting results shown in Figure 5.2, it is difficult to determine which model performs better. Both models produce reasonably smooth outputs, but the OutpaintingDDPM remains slightly noisy, and the OutpaintingWGAN still exhibits contrast between the generated and original regions.

Quantitative results: As reflected in Table 5.2, the results for outpainting are consistent with those from the inpainting task. The OutpaintingDDPM has a better FID, but both models show similar SSIM and L1 scores.


(a) InpaintingWGAN (masked, generated, target).

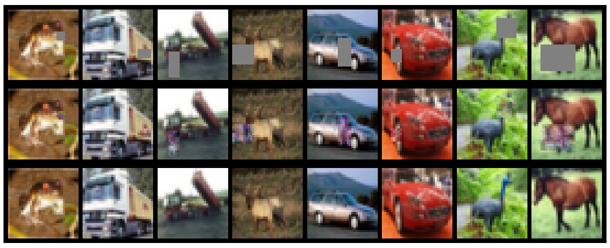
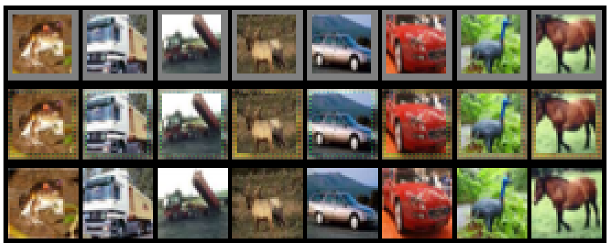

(b) InpaintingDDPM (masked, generated, target).

Figure 5.1: Inpainting results comparison: WGAN vs. DDPM.

Model	**FID** ↓	**SSIM** ↑	**L1** ↓
WGAN	38.51	0.86	407.20
DDPM	22.96	0.85	396.23

Table 5.1: Inpainting evaluation metrics.


(a) OutpaintingWGAN (masked, generated, target).

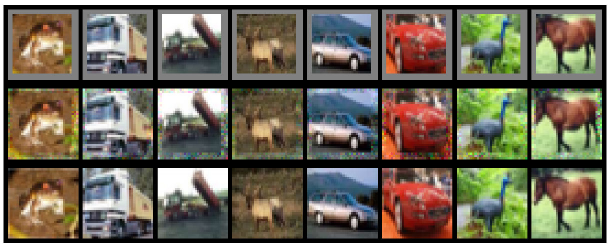

(b) OutpaintingDDPM (masked, generated, target).

Figure 5.2: Outpainting results comparison: WGAN vs. DDPM.

Model	**FID** ↓	**SSIM** ↑	**L1** ↓
WGAN	42.39	0.85	37.18
DDPM	28.64	0.84	32.36

Table 5.2: Outpainting evaluation metrics.

Chapter 6

Discussion

It is important to note that several factors may introduce bias into our analysis.

6.0.1 Dataset Bias

First, we used random masks for the Inpainting task, which may introduce noise into our evaluation, as the models are exposed to masks of varying difficulty.

Additionally, we only tested on the CIFAR10 dataset, yet data distribution plays a crucial role in model performance. Different datasets may be better suited to one model over the other, potentially influencing the results.

6.0.2 Computational Cost vs. Stability

During our experiments, we observed a trade-off between computational cost and training stability. As stated before, both models were designed to have a similar number of parameters.

However, the DDPM takes significantly longer to train and is much slower during inference. This makes it less practical in scenarios where time efficiency is critical. Furthermore, implementing a DDPM is more complex, involving several design decisions, such as how to encode time embeddings, where to inject them, and how to define the noise schedule.

On the other hand, tuning a WGAN also presents challenges, as it is highly sensitive to hyperparameters and may require careful balancing to avoid instability or mode collapse. To guide the training, we introduced a reconstruction loss term in the WGAN objective. We did not incorporate an equivalent term in the DDPM loss, which also introduces a bias in favor of the WGAN.

Chapter 7

Conclusion

In this work, we compared Wasserstein Generative Adversarial Networks (WGANs) and Denoising Diffusion Probabilistic Models (DDPMs) on three conditional image restoration tasks: inpainting, outpainting, and super-resolution. We evaluated both models qualitatively and quantitatively using standard metrics such as FID, SSIM, and L1 reconstruction error.

Our results show that while both models achieve similar pixel-wise and structural accuracy, DDPMs consistently outperform WGANs in terms of FID, suggesting a better alignment with the true data distribution. However, DDPMs occasionally generate samples with visible noise or blur, particularly in fine details. WGANs, despite slightly lower quantitative scores, often produce sharper and visually appealing outputs, though they can suffer from contrast inconsistencies and artifacts.

That said, drawing a perfectly fair comparison between the two families of models remains challenging. Differences in architectural complexity, training dynamics, and the specific choice of loss functions all introduce bias. Moreover, performance is highly sensitive to the dataset and task, meaning that other architectures or training strategies could lead to different conclusions.

Ultimately, the choice between WGANs and DDPMs should be guided by the specific constraints and goals of the application. DDPMs are preferable when distributional realism is critical and resources are sufficient, while WGANs remain a strong baseline in low-latency or resource-constrained environments.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. 2017.

[2] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. 2021.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. 2014.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. 2017.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. 2020.

[6] Alex Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. 2021.

[7] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. 2021.

[8] Lilian Weng. From gan to wgan. https://lilianweng.github.io/posts/2017-08-20-gan/, 2017.

[9] Lilian Weng. What are diffusion models? https://lilianweng.github.io/posts/2021-07-11-diffusion-models/, 2021.