GAN Inversion: Summary and Optimization-Based Implementation Review

Assaf Zimand

October 21, 2025

Abstract

Generative Adversarial Networks (GANs) have demonstrated remarkable capabilities in synthesizing realistic images from latent representations. The process of recovering a latent code that reproduces a given real image, known as *GAN inversion*, is the focus of this summary. It provides an overview of the principal approaches to GAN inversion, based on the survey article *"GAN Inversion: A Survey"* [1], with a detailed mathematical treatment of optimization-based inversion, the method implemented in this project. We discuss the underlying objective, the role of latent spaces (Z, W, W^+), loss formulations, and initialization strategies. Finally, we analyze the trade-offs among different inversion paradigms and highlight the strengths of optimization-based methods within this landscape.

1 Overview

GAN inversion seeks to map a real image x into a latent representation z^* such that the generated image $G(z^*)$ closely reconstructs x. Given a pretrained generator G, inversion enables reconstruction, semantic editing, and understanding of the latent space structure. Since GANs were introduced, subsequent works such as StyleGAN and its variants have provided structured latent spaces that enable powerful inversion and editing capabilities.

2 Implementation Review

The accompanying project repository implements only the *optimization-based inversion* approach. It performs several controlled experiments comparing different:

- **Initialization methods:** mean-w and encoder-based.
- **Latent spaces:** W, and W^+.
- **Loss functions:** L2 and perceptual (LPIPS).

The implementation uses pretrained models from StyleGAN2-FFHQ-128 and Encoder4Editing. The full source code is publicly available at:

`https://github.com/assafzimand/GAN_Inversion`

Users can refer to the README file for detailed usage instructions, experimental settings, and visual comparisons of results under different parameters.

3 Summary of GAN Inversion Methods

3.1 Learning-Based Inversion

Learning-based inversion methods train an encoder network $E(x)$ to approximate the inverse mapping of G. Once trained, inversion can be performed in a single forward pass: $z^* = E(x)$. These methods are computationally efficient but often achieve lower reconstruction fidelity.

3.2 Optimization-Based Inversion

Optimization-based approaches directly minimize a reconstruction loss between the real image x and the generated image $G(z)$, optimizing over the latent variable z. This produces high-quality results but requires iterative gradient-based optimization.

3.3 Hybrid Methods

Hybrid methods combine encoders with optimization refinements, leveraging encoder efficiency for initialization and the accuracy of optimization for refinement. As reviewed in [1], these approaches can be divided into two main categories:

- **Encoder-assisted optimization:** where an encoder provides an initial latent code (e.g., $E(x)$), which is then refined via optimization in the latent space.
- **Iterative refinement frameworks:** where inversion proceeds through multiple encoder passes or residual corrections, as in ReStyle or HyperInverter.

Hybrid methods therefore reduce over-optimization and maintain realistic reconstructions, while improving speed compared to pure optimization and preserving high reconstruction fidelity. However, they still face challenges in generalizing to unseen domains.

3.4 Discussion and Trade-Offs

Optimization-based inversion offers excellent reconstruction quality but at the cost of computational time and sensitivity to initialization. Learning-based methods are faster but less precise, while hybrid methods combine both advantages. The implementation experiments in the linked repository demonstrate these effects empirically and visually.

4 Applications of GAN Inversion

4.1 Latent Space Exploration and Editing

After inversion, latent codes can be manipulated along interpretable directions to achieve edits such as pose, expression, or age modifications.

4.2 Applications

GAN inversion supports a wide range of tasks, including image editing, restoration, domain translation, and content manipulation. As reviewed in [1], domain-specific applications include face editing and manipulation, super-resolution, inpainting, and image de-blurring. Inversion also provides insights into the geometry of the latent space, enabling interpretable manipulation and semantic understanding of generative models.

5 Mathematical Foundation of Optimization-Based Inversion

5.1 Objective Function

The optimization-based inversion aims to find a latent vector z^* that minimizes a *loss function* measuring the similarity between the generated image and the target:

$$z^* = \arg\min_z \ell(G(z), x) \tag{1}$$

Here, $\ell(G(z), x)$ is a loss function defined over the image or feature space. This loss can take various forms depending on the chosen image similarity metric, including pixel-wise distances such as L_1 or L_2, perceptual or feature-based losses, and the LPIPS metric that evaluates perceptual similarity using deep features. Additional constraints, such as identity preservation or semantic consistency, can also be incorporated when appropriate.

In practice, the optimization is typically performed over the intermediate latent variable w or its layer-wise extension w^+, rather than directly over z. This design leverages the smoother loss landscape and better disentanglement properties of the W space introduced by the mapping network $f : Z \to W$. Because the inversion objective is non-convex, the choice of optimizer also plays a critical role. Common optimizers include gradient-based methods such as **Adam** and **L-BFGS**, as well as more advanced or gradient-free approaches such as **Hamiltonian Monte Carlo (HMC)** and **Covariance Matrix Adaptation (CMA)**. Different optimizers can lead to notable variations in reconstruction fidelity and convergence speed.

5.2 Latent Spaces Z, W, and W^+

In StyleGAN, the input latent space Z is sampled from a normal prior $\mathcal{N}(0, I)$. A mapping network $f : Z \to W$ transforms z into an intermediate latent vector w:

$$w = f(z) \tag{2}$$

The extended space W^+ allows a different latent vector w_i for each generator layer:

$$W^+ = \{(w_1, w_2, \ldots, w_L) \mid w_i \in W\} \tag{3}$$

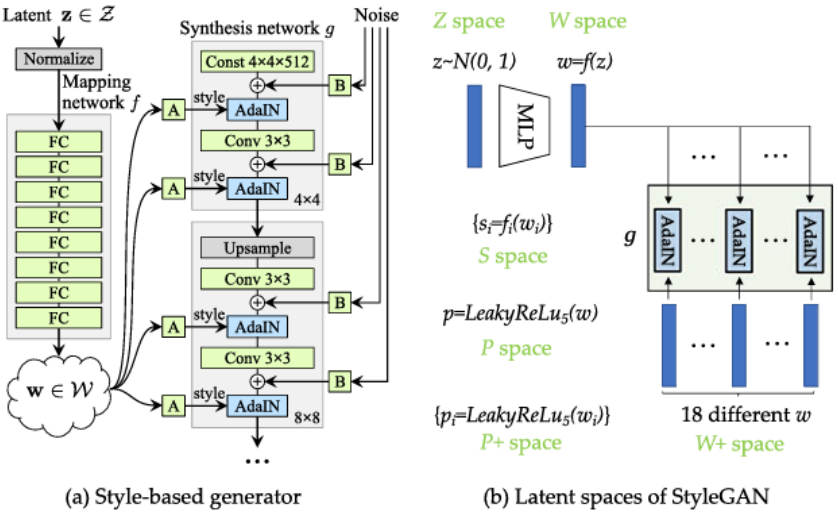
where L denotes the number of layers in the generator. This layer-wise flexibility provides greater reconstruction accuracy but may reduce the disentanglement of semantic attributes compared to the more compact W space, meaning that latent directions become less semantically independent and edits in the latent space may affect multiple features simultaneously. However, this reduced disentanglement does not negatively impact the goal of image reconstruction, which often benefits from the additional flexibility provided by the W^+ space.

5.3 Comparative Discussion

- Z: compact but less disentangled.
- W: more interpretable and semantically meaningful.
- W^+: layer-wise flexibility for better reconstruction, but reduced edit consistency.

Beyond W^+, several extended latent spaces have been proposed, such as P^+, S, and hierarchical spaces that jointly optimize both mapping and synthesis layers (as in StyleGAN3).

These variants extend the same underlying idea of improving reconstruction quality through increased latent flexibility, as discussed in [1].



(a) Style-based generator (b) Latent spaces of StyleGAN

Figure 1: Schematic illustration of the mapping network and latent spaces $Z \to W \to W^+$.

5.4 Loss Terms

The reconstruction loss usually combines pixel-level and perceptual components to measure the similarity between the generated image $G(z)$ and the target image x.

Pixel-level loss (L2):

$$\ell_{\text{L2}} = \|G(z) - x\|_2^2 \tag{4}$$

This loss enforces direct pixel-wise correspondence, ensuring low-level accuracy in color and structure. However, relying solely on L2 often leads to blurry reconstructions because it does not capture perceptual or semantic similarity.

Perceptual loss (LPIPS):

$$\ell_{\text{perc}} = \|\phi(G(z)) - \phi(x)\|_2^2 \tag{5}$$

where ϕ denotes deep feature embeddings extracted from a pretrained network (e.g., VGG). This loss evaluates perceptual similarity in feature space rather than pixel space, helping to preserve texture and high-level visual details. It aligns the generated image with the target in terms of how they are perceived by a neural network trained on natural images.

Combined objective:

$$\ell(G(z), x) = \lambda_1 \ell_{\text{L2}} + \lambda_2 \ell_{\text{perc}} \tag{6}$$

In practice, these two terms are combined to balance structural accuracy and perceptual realism. The coefficients λ_1 and λ_2 control their relative influence during optimization.

These specific terms are shown explicitly because they are the ones implemented in the project's experiments. Beyond these, the survey article [1] discusses several other losses used in different inversion frameworks, including:

- **Identity loss**, which compares deep facial features (e.g., from ArcFace or VGGFace) to ensure the reconstructed face preserves the same identity as the input.
- **Feature-matching or semantic loss**, which is conceptually similar to LPIPS but differs in that it compares intermediate feature representations from pretrained or task-specific networks (such as the discriminator or encoder), aiming to maintain semantic alignment and structural consistency rather than perceptual similarity alone.
- **Adversarial loss**, occasionally used to keep reconstructed images within the generator's learned distribution and prevent unrealistic artifacts.

These additional terms are often task-dependent, improving perceptual or semantic fidelity in specific domains, whereas pixel-based and perceptual (LPIPS) losses together form a standard baseline for high-quality image reconstruction in most inversion frameworks.
In addition to these reconstruction terms, optimization-based inversion often incorporates regularization terms to constrain the latent code within the valid manifold of the generator. Common choices include a latent norm penalty $\|w-\bar{w}\|_2^2$ or prior regularization on z or w, which help prevent unrealistic artifacts and maintain consistency with the generator's prior distribution [1].

5.5 Initialization Strategies

Different initialization strategies affect convergence and reconstruction quality:

- **Mean-w:** Initialize with the average latent vector $\bar{w}$ computed as $\bar{w} = E_{z\sim\mathcal{N}(0,I)}[\, f(z)\,]$.
- **Random (via $Z \to W$):** Sample $z \sim \mathcal{N}(0, I)$ and set $w = f(z)$. For W^+, either broadcast this single w to all layers $(w, \ldots, w)$, or sample multiple z_i and set $w_i = f(z_i)$ per layer.
- **Encoder-based:** Use a pretrained encoder $E(x)$ to produce an initial code (e.g., $w_0 = E(x)$ or $w_0^+ = E(x)$), then refine by optimization.

5.6 Evaluation Metrics

Quantitative evaluation of inversion quality commonly relies on pixel-level and perceptual similarity metrics. The survey [1] highlights the most frequently used ones:

- **PSNR (Peak Signal-to-Noise Ratio):** measures pixel fidelity; higher values indicate better reconstruction.
- **SSIM (Structural Similarity Index):** captures perceived structural consistency between images.
- **LPIPS (Learned Perceptual Image Patch Similarity):** evaluates perceptual distance in deep feature space.

Together, these metrics reflect the balance between visual realism and reconstruction accuracy.

5.7 Datasets and Domain Gap

Most GAN inversion studies evaluate performance using high-quality face datasets such as FFHQ and CelebA-HQ. However, a significant domain gap exists when inverting out-of-domain images, such as artworks, cartoons, or hand-drawn sketches, using generators trained on natural photos. This mismatch remains a major challenge, motivating research into domain adaptation and cross-domain inversion techniques.

6 Conclusion

This summary reviewed the main families of GAN inversion methods, with a mathematical focus on optimization-based inversion. By formulating inversion as an optimization problem in latent space, we achieve accurate reconstructions and gain insights into generative models. Future work may integrate adaptive losses and improved initialization to enhance both speed and fidelity. Despite substantial progress, open challenges remain. Key limitations include the high computational cost of optimization-based methods, limited generalization across image domains, and the inherent trade-off between reconstruction fidelity and editability. Recent works also explore diffusion-based inversion that may achieve both precision and flexibility.

References

[1] Z. Yang, Y. Xu, H. He, S. Liu, and X. Li, *GAN Inversion: A Survey*, arXiv preprint arXiv:2301.08281, 2023.

[2] H. Ajarrar, "StyleGAN2-FFHQ-128 pretrained model," Available at: `https://huggingface.co/hajar001/stylegan2-ffhq-128`.

[3] O. Tov *et al.*, "Encoder4Editing: Official PyTorch implementation," Available at: `https://github.com/omertov/encoder4editing`.

Appendix: Project Repository

The full implementation accompanying this summary is available at:

`https://github.com/assafzimand/GAN_Inversion`

It contains the source code for optimization-based inversion experiments, including comparisons of initialization methods, latent spaces, and loss functions, as well as example results and usage instructions in the README file.