CHAPTER

Fundamentals of generative adversarial networks

3

Songtao Lu, Pei Tang, and Zihao Zheng
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong Special Administrative Region

3.1 Introduction to GANs

Generative Adversarial Networks (GANs), first introduced by Ian Goodfellow et al. in 2014 [6], represent a transformative class of generative models in machine learning. The fundamental concept involves training two neural networks, the generator and the discriminator, in a competitive framework. The generator aims to produce synthetic data that closely resemble real samples, while the discriminator seeks to distinguish between real and generated data. Through this iterative competition, GANs learn to approximate complex, high-dimensional data distributions and are capable of generating realistic outputs across various modalities, including images, audio, and text.

Since their introduction, GANs have rapidly evolved through a series of architectural and algorithmic innovations designed to address challenges in training stability, convergence behavior, and output quality. For example, convolutional architectures have facilitated stable and scalable image synthesis. Approaches based on optimal transport, such as the Wasserstein GAN, have been developed to overcome limitations associated with divergence-based loss functions. Other advances include controllable generation, style-conditioned synthesis, and unsupervised domain translation, all of which have extended the practical reach of GAN-based models across diverse tasks.

Beyond their success in traditional machine learning domains, GANs have also shown considerable promise in wireless communications. By learning to replicate the statistical characteristics of wireless channels or traffic patterns, GANs can improve the realism of simulations and enable effective data augmentation. Their capacity to generate synthetic yet representative training data is particularly valuable in edge scenarios with limited access to labeled samples. Moreover, training strategies rooted in nonconvex min-max optimization, inspired by the GAN framework, have been employed to model adverse conditions such as jamming or interference. These developments contribute to enhancing the robustness and adaptability of

Generative Learning for Wireless Communications. https://doi.org/10.1016/B978-0-44-341497-8.00010-9

communication systems [14]. Collectively, they position GANs as a powerful tool not only for generative modeling but also for enabling intelligent, data-driven design and optimization in modern wireless networks [22].

3.2 Principles of GAN architecture and algorithms

3.2.1 Formulation and objective of GANs

GANs consist of two neural networks that are trained simultaneously in a competitive setting:

- *Discriminator* D: A binary classifier that receives data samples and predicts whether they are real (drawn from the true data distribution) or fake (produced by the generator). Its goal is to correctly distinguish real data from generated data.
- *Generator* G: A neural network that maps a latent variable $z \sim p_z(z)$, drawn from a known prior distribution p_z (e.g., standard Gaussian $\mathcal{N}(0, I_d)$ or uniform distribution on $[-1, 1]^d$), to a data sample $\tilde{x} = G(z)$. The generator aims to produce samples that resemble the real data well enough to fool the discriminator.

The true data distribution is denoted by $p_{\text{data}}(x)$, from which real samples x are drawn. The distribution of generated data, denoted $p_g(x)$, is implicitly defined by the transformation of latent variables $z \sim p_z$ through the generator:

$$x = G(z), \quad z \sim p_z \quad \Rightarrow \quad x \sim p_g.$$

Although p_g is not known in closed form, it is accessible through samples generated by G.

The interaction between D and G is framed as a two-player min-max game with the following objective:

$$\min_G \max_D f(D, G) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D(x)\right] + \mathbb{E}_{z \sim p_z} \left[\log(1 - D(G(z)))\right].$$

This loss originates from the binary cross-entropy used in classification. In the GAN setting:

- The discriminator is trained to assign high probability to real samples and low probability to generated ones.
- The generator is trained to produce samples that maximize the discriminator's output, effectively trying to minimize the loss $\log(1 - D(G(z)))$.

Cross-entropy is a natural loss for binary classification, as it penalizes discrepancies between predicted probabilities and true binary labels. In GANs, minimizing this loss encourages the discriminator to make accurate predictions, while pushing the generator to produce increasingly realistic outputs.

Interpretation via Jensen–Shannon Divergence. Although the GAN objective is based on cross-entropy, Goodfellow et al. [6] showed that, under an optimal

discriminator, this objective is equivalent to minimizing a statistical distance between distributions.

For a fixed generator G, the value function can be written as

$$\begin{aligned} V(D \mid G) &= \mathbb{E}_{x\sim p_{\text{data}}}\big[\log D(x)\big] + \mathbb{E}_{x\sim p_g}\big[\log\big(1 - D(x)\big)\big] \\ &= \int p_{\text{data}}(x)\,\log D(x)\,dx + \int p_g(x)\,\log\big(1 - D(x)\big)\,dx. \end{aligned} \tag{3.1}$$

For each fixed x, let $y := D(x) \in (0, 1)$ and define

$$\phi_x(y) = p_{\text{data}}(x)\,\log y + p_g(x)\,\log(1 - y).$$

Maximizing $\phi_x(y)$ with respect to y amounts to setting its derivative to zero:

$$\frac{d\phi_x}{dy} = \frac{p_{\text{data}}(x)}{y} - \frac{p_g(x)}{1 - y} = 0 \implies y^\star = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}.$$

Moreover, ϕ_x is strictly concave since

$$\frac{d^2\phi_x}{dy^2} = -\frac{p_{\text{data}}(x)}{y^2} - \frac{p_g(x)}{(1 - y)^2} < 0,$$

so $y^\star$ is the unique maximizer. Therefore, the optimal discriminator is

$$D^\star(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}.$$

Note that

$$\begin{aligned} \log D^\star(x) &= \log p_{\text{data}}(x) - \log\big(p_{\text{data}}(x) + p_g(x)\big), \\ \log\big(1 - D^\star(x)\big) &= \log p_g(x) - \log\big(p_{\text{data}}(x) + p_g(x)\big). \end{aligned}$$

Plugging these into Eq. (3.1) yields

$$\begin{aligned} V(D^\star \mid G) = &\int p_{\text{data}}(x)\,\log p_{\text{data}}(x)\,dx + \int p_g(x)\,\log p_g(x)\,dx \\ &- \int \big(p_{\text{data}}(x) + p_g(x)\big)\,\log\big(p_{\text{data}}(x) + p_g(x)\big)\,dx. \end{aligned}$$

Introduce the mixture $M(x) = \frac{1}{2}\big(p_{\text{data}}(x) + p_g(x)\big)$. Then

$$\big(p_{\text{data}} + p_g\big)\,\log\big(p_{\text{data}} + p_g\big) = 2M\,\log(2M) = 2M\big(\log 2 + \log M\big).$$

Therefore,

$$
\begin{aligned}
& V(D^\star \mid G) \\
&= \int p_{\text{data}} \log p_{\text{data}}\, dx \;+\; \int p_g \log p_g\, dx \;-\; 2\int M\big(\log 2 + \log M\big)\, dx \\
&= \Big[\int p_{\text{data}} \log \tfrac{p_{\text{data}}}{M}\, dx\Big] + \Big[\int p_g \log \tfrac{p_g}{M}\, dx\Big] \;-\; 2\log 2 \underbrace{\int M\, dx}_{=1},
\end{aligned}
$$

where we used the identity

$$
\int p_{\text{data}} \log M\, dx + \int p_g \log M\, dx = 2\int M \log M\, dx.
$$

By the definition of the Kullback–Leibler divergence,

$$
V(D^\star \mid G) = \mathrm{KL}(p_{\text{data}} \,\|\, M) + \mathrm{KL}\big(p_g \,\|\, M\big) - 2\log 2.
$$

Hence,

$$
V(D^\star \mid G) \;=\; -\log 4 \;+\; 2\,\mathrm{JS}\big(p_{\text{data}} \,\|\, p_g\big).
$$

Here JS denotes the Jensen–Shannon divergence, defined for distributions P, Q by

$$
\mathrm{JS}(P \,\|\, Q) \;=\; \tfrac{1}{2}\,\mathrm{KL}(P \,\|\, M) \;+\; \tfrac{1}{2}\,\mathrm{KL}(Q \,\|\, M), \quad M \;=\; \tfrac{1}{2}(P+Q).
$$

Therefore, the generator's problem $\min_G V(D^\star \mid G)$ is equivalent to minimizing $\mathrm{JS}(p_{\text{data}} \,\|\, p_g)$. The minimum is attained at $p_g = p_{\text{data}}$, where $\mathrm{JS} = 0$ and the value is $-\log 4$. Thus, GAN training can be interpreted as implicitly minimizing the Jensen–Shannon divergence between the real data distribution p_{data} and the generator distribution p_g.

3.2.2 Variants of GANs

Vanilla GANs have achieved significant success in various AI applications. However, due to their inherent limitations, such as a lack of control over generated outputs and instability during training, their applicability remains restricted. To address these issues, several extensions and variants of GANs have been proposed. Two notable examples are Conditional GANs (CGANs) and Wasserstein GANs (WGANs), which are introduced below.

Conditional GANs

Standard GANs generate data solely from random noise, offering little control over the output. In many real-world applications, however, it is desirable to guide the generation process using auxiliary information—for example, generating images of a specific digit, object category, or based on a text description. CGANs address this limitation by incorporating conditioning variables into both the generator and the discriminator, enabling more targeted and controllable data generation [16].

In a CGAN, both the generator G and the discriminator D are conditioned on additional input y, which may represent class labels, textual descriptions, or other modalities. Conditioning is achieved by feeding y into both the generator and the discriminator as part of their input layers, allowing them to incorporate the auxiliary information during training and generation.

The objective function extends the original GAN min-max game to include the conditioning variable:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x \mid y)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z \mid y)))].$$

This conditional framework enables the generator to learn the class-conditional distribution $p(x \mid y)$, rather than the marginal distribution $p(x)$, thereby enhancing controllability and improving sample diversity within each class. It also supports label-guided synthesis and data augmentation in downstream tasks.

Wasserstein GANs (WGANs)

Another major limitation of vanilla GANs stems from the behavior of the Jensen–Shannon (JS) divergence: it is only sensitive to differences in regions where the two distributions overlap. When the real data distribution p_{data} and the generator distribution p_g have disjoint or nearly disjoint support, as is common in the early stages of training, the JS divergence becomes constant, resulting in zero gradients. In this regime, the discriminator quickly saturates, assigning confident predictions to both real and fake samples, while the generator receives little to no informative feedback. This leads to vanishing gradients and, consequently, unstable or stalled training dynamics.

These limitations motivate alternative divergence measures or distances that yield meaningful gradients even when the supports of the distributions are disjoint. One such approach is the WGAN, which replaces the JS divergence with the Earth-Mover (Wasserstein-1) distance, resulting in smoother optimization and improved convergence behavior. The Wasserstein-1 distance between the real data distribution p_{data} and the generator distribution p_g is defined as:

$$W(p_{\text{data}}, p_g) = \inf_{\gamma\in\Pi(p_{\text{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\|\right],$$

where $\Pi(p_{\text{data}}, p_g)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are p_{data} and p_g, respectively. Intuitively, $W(p_{\text{data}}, p_g)$ represents the minimum cost of transporting the probability mass of p_g to match p_{data}, with cost measured by the ground distance $\|x - y\|$.

This primal form is computationally intractable in high dimensions. To make optimization feasible, WGAN leverages the Kantorovich–Rubinstein duality, which states:

$$W(p_{\text{data}}, p_g) = \sup_{\|D\|_L\leq 1} \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{x\sim p_g}[D(x)],$$

where the supremum is taken over all real-valued 1-Lipschitz functions D. Here, the joint distribution γ disappears because the optimization is now over functions D rather than couplings γ, and the marginals p_{data} and p_g are used directly in the expectations.

In this formulation, D becomes a *critic* (rather than a discriminator), mapping inputs to real numbers without the probabilistic constraint. The WGAN training objective thus becomes:

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

where p_z is the latent prior and $\mathcal{D}$ denotes the set of all 1-Lipschitz functions.

To enforce the Lipschitz constraint, the original WGAN applied weight clipping to restrict the critic's parameters within a bounded range. However, this can lead to poor capacity and training instability. An improved variant, WGAN-GP [7], introduces a soft constraint via a gradient penalty:

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

where $\hat{x}$ is sampled uniformly along straight lines between real and generated samples, and λ is a regularization coefficient.

By directly minimizing the Wasserstein-1 distance $W(p_{\text{data}}, p_g)$, WGANs provide a well-behaved loss function that yields non-vanishing gradients even when p_{data} and p_g lie on low-dimensional manifolds. This leads to more stable training dynamics, improved convergence, and higher-quality sample generation.

In summary, both CGANs and WGANs extend the original GAN framework by addressing key practical limitations. CGANs enable controlled generation by incorporating auxiliary information, allowing the model to produce samples conditioned on specific inputs. WGANs, on the other hand, replace the adversarial divergence with the Wasserstein distance, offering improved optimization dynamics and greater training stability. These advancements have become foundational in the development of modern GAN architectures, particularly for complex tasks such as text-to-image synthesis, domain adaptation, and high-resolution image generation.

3.2.3 Min-max optimization and training algorithms for GANs

Mathematically, the training objective of GANs can be formulated as the following min-max optimization problem:

$$\min_{\theta_G} \max_{\theta_D} f(\theta_G, \theta_D), \tag{3.2}$$

where the optimization variables θ_G and θ_D represent the model parameters of the generator and discriminator, respectively.

A widely recognized notion of optimality for this min-max optimization problem is the *Nash equilibrium* from game theory, which describes a situation in which no player can improve their outcome by unilaterally changing their strategy while the other player's strategy remains fixed. Formally, a point $(\theta_G^\star, \theta_D^\star)$ is a Nash equilibrium of a function f if

$$f(\theta_G^\star, \theta_D) \leq f(\theta_G^\star, \theta_D^\star) \leq f(\theta_G, \theta_D^\star),$$

where the feasible sets are compact. That is, $\theta_G^\star$ is a global minimum of $f(\cdot, \theta_D^\star)$ when the *discriminator*'s parameters are fixed, and $\theta_D^\star$ is a global maximum of $f(\theta_G^\star, \cdot)$ when the *generator*'s parameters are fixed.

A growing body of research has focused on solving the min-max optimization problem (3.2), with applications to GANs [13] and wireless communications [15]. The most straightforward approach to solving (3.2) is the *gradient descent-ascent* (GDA) algorithm, which alternates between a gradient descent step on the generator parameters θ_G and a gradient ascent step on the discriminator parameters θ_D. The update rules are given by

$$\theta_G^{t+1} = \theta_G^t - \alpha_G \nabla_{\theta_G} f(\theta_G^t, \theta_D^t), \tag{3.3a}$$
$$\theta_D^{t+1} = \theta_D^t + \beta_D \nabla_{\theta_D} f(\theta_G^t, \theta_D^t), \tag{3.3b}$$

where α_G and β_D denote the learning rates (or step sizes) for the generator and discriminator, respectively, and t indicates the iteration index. It can be seen that the generator and discriminator update their parameters simultaneously using gradient descent and ascent steps. Under certain regularity conditions on the objective function, such as convexity in θ_G and concavity in θ_D, the GDA algorithm can be shown to converge to a Nash equilibrium at a sublinear rate [9].

An important variant of the GDA algorithm is the *Optimistic Gradient Descent Ascent* (OGDA) algorithm [3], which improves the stability and convergence behavior of min-max optimization problems. Unlike GDA, which uses only current gradients for updating the generator and discriminator parameters, OGDA incorporates an *extra-gradient* or momentum-like term that leverages both the current and previous gradients to anticipate the opponent's move. Specifically, the update rule of OGDA is as follows:

$$\theta_G^{t+1} = \theta_G^t - 2\alpha_G \nabla_{\theta_G} f(\theta_G^t, \theta_D^t) + \alpha_G \nabla_{\theta_G} f(\theta_G^{t-1}, \theta_D^{t-1}), \tag{3.4a}$$
$$\theta_D^{t+1} = \theta_D^t + 2\beta_D \nabla_{\theta_D} f(\theta_G^t, \theta_D^t) - \beta_D \nabla_{\theta_D} f(\theta_G^{t-1}, \theta_D^{t-1}), \tag{3.4b}$$

where α_G and β_D are again the step sizes, and $(\theta_G^{t-1}, \theta_D^{t-1})$ are the parameters from the previous iteration.

The key idea behind OGDA is to include *optimism* in the gradient steps, effectively forecasting the next gradient direction based on past behavior. This method is particularly effective in min-max optimization settings where the interaction between players introduces rotational dynamics that can hinder convergence. OGDA mitigates

Algorithm 1 Two-time scale gradient descent ascent training.

Require: generator step size $\alpha_G > 0$, discriminator/critic step size $\beta_D > 0$; critic steps $n \geq 1$; total iterations T; data distribution p_{data}; latent prior p_z

Initialize parameters θ_G^1, θ_D^0
for $t = 1, \ldots, T$ **do**
  for $s = 0, \ldots, n-1$ **do** ▷ inner ascent to approximate $\max_{\theta_D} f(\theta_G^t, \theta_D)$
    Sample minibatches $\{x_i\}_{i=1}^m \sim p_{\text{data}}$ and $\{z_i\}_{i=1}^m \sim p_z$
    Estimate $g_D^s \approx \nabla_{\theta_D} f(\theta_G^t, \theta_D^s)$ using the sampled minibatches
    $\theta_D^{s+1} \leftarrow \theta_D^s + \beta_D\, g_D^s$
  end for
  Sample a minibatch $\{z_i\}_{i=1}^m \sim p_z$ ▷ freeze θ_D^n during the generator step
  Estimate $g_G^t \approx \nabla_{\theta_G} f(\theta_G^t, \theta_D^n)$ using the sampled minibatch
  $\theta_G^{t+1} \leftarrow \theta_G^t - \alpha_G\, g_G^t$
  $\theta_D^0 \leftarrow \theta_D^n$
end for

such oscillations and accelerates convergence by stabilizing the trajectory around saddle points or equilibria. Theoretical results show that, under certain smoothness and monotonicity conditions, OGDA achieves greater stability than standard GDA [3].

In practice, GANs are trained with a double-loop, two-time scale saddle-point routine based on $F(\theta_G) = \max_{\theta_D} f(\theta_G, \theta_D)$. The discriminator or critic (parameters θ_D) is updated for several inner steps to approximately maximize f, and then the generator (parameters θ_G) is updated once to decrease F. By Danskin's theorem, during the generator step we freeze the discriminator, treating the current maximizer $\theta_D^\star(\theta_G) \in \arg\max_{\theta_D} f(\theta_G, \theta_D)$ as constant, so the generator gradient is obtained by the chain rule through the discriminator's input. Compared with a single-loop scheme, this structure yields a more reliable $\nabla_{\theta_G} F$, i.e.,

$$\nabla_{\theta_G} F(\theta_G) = \nabla_{\theta_G}\left[\max_{\theta_D} f(\theta_G, \theta_D)\right] \approx \nabla_{\theta_G} f\left(\theta_G, \theta_D^\star(\theta_G)\right),$$

because additional discriminator steps move θ_D closer to a local maximizer and make $\nabla_{\theta_G} D(G(z))$ more informative. Simultaneous gradient descent and ascent is prone to cycling and rotations; separating the time scales by using more or stronger discriminator updates, or a larger discriminator step size, damps these dynamics and improves convergence.

Additionally, as datasets are large, practical training is stochastic: each iteration (i) draws mini-batches $x_i \sim p_{\text{data}}$ and $z_i \sim p_z$ where $i \in [m]$, (ii) performs $n \geq 1$ discriminator updates to increase the chosen adversarial objective (for example, the JS score or, for WGAN, the critic gap) with any required regularization (for example, the WGAN-GP gradient penalty), and then (iii) performs one generator update using the frozen discriminator signal (for example, WGAN's $-\mathbb{E}\, D_{\theta_D}(G_{\theta_G}(z))$). Two-time scale hyperparameters, such as n, learning rates, and regularization strength, govern

stability and speed; diagnostics such as the critic gap, the gradient-penalty norm, and sample quality metrics track progress. This training strategy applies broadly across GAN variants. The detailed implementation of the double-loop method appears in Algorithm 1.

3.2.4 Practical risks and limitations of GANs

Despite their remarkable success in generating high-fidelity data, a major drawback of GANs lies in their lack of robustness and interpretability. A primary reason is that, in practice, the loss function is typically nonconvex, since both the generator and discriminator are neural networks parameterized by θ_G and θ_D. This nonconvexity renders the problem of finding a Nash equilibrium NP-hard. Another major challenge arises from the opposing objectives of the two players, which often lead to instability during optimization.

To address this, the notion of stationarity commonly adopted in the nonconvex optimization community is used as a surrogate for Nash optimality. Assume that f is differentiable. A point $(\theta_G^\star, \theta_D^\star)$ that satisfies the first-order stationarity conditions

$$\nabla_{\theta_G} f(\theta_G^\star, \theta_D^\star) = 0 \quad \text{and} \quad \nabla_{\theta_D} f(\theta_G^\star, \theta_D^\star) = 0$$

is called a *quasi-Nash equilibrium* [18]. Although numerous studies have proposed new optimization algorithms and theoretical analyses to tackle nonconvex–nonconcave min–max problems, these approaches typically rely on specific problem structures [11] or additional regularity conditions [12].

In many cases, small perturbations in the data or initialization can lead to significant deviations in the quality of the generated outputs, posing serious challenges in safety-critical applications. These factors often lead to instability during training, most notably manifesting as mode collapse. In addition to robustness concerns, GANs also suffer from poor interpretability. Unlike models with well-defined likelihood functions or attention mechanisms, GANs provide little insight into how specific features of the input influence the generated outputs. This opacity limits our ability to diagnose failure modes or develop theoretical guarantees, thereby motivating ongoing research into interpretable GAN architectures, explainable training dynamics, and robust regularization techniques.

3.3 Applications of GANs

3.3.1 Image generation and synthesis

Experimental results are conducted on the MNIST dataset [10] using three representative variants of GANs: Vanilla GAN [6], CGAN [16], and WGAN [1,7]. MNIST is a widely used benchmark dataset consisting of ten classes of handwritten digits and contains a total of 60,000 grayscale training images.

In each training iteration, real images are drawn from the MNIST training set, while synthetic images are generated by sampling latent vectors from a Gaussian distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, where $\mathbf{I}_d$ denotes the d-dimensional identity matrix.

All three GAN models adopt an identical neural network architecture. The discriminator comprises four fully connected layers with ReLU activation functions, and the generator consists of four fully connected layers incorporating batch normalization and ReLU activations. A Tanh activation is applied at the output layer of the generator to constrain the pixel intensities to the range $[-1, 1]$. In the case of CGAN and WGAN, a ten-dimensional one-hot vector representing the class label is assigned to each image and is used as a conditional input to both the generator and the discriminator.

To improve training stability in WGAN, a gradient penalty term is added to the discriminator loss. Additionally, the generator is updated once every five discriminator updates. During each training iteration, 128 real images and 128 latent vectors are sampled. Model parameters are optimized using the Adam optimizer with a learning rate of 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, for a total of 50,000 iterations.

Throughout training, both the loss trajectories and generated outputs are recorded. Fig. 3.1(a) illustrates the convergence behavior of the Vanilla GAN, while Fig. 3.1(b) shows representative samples generated at various stages. As training progresses and the loss values stabilize, the quality of the generated images improves significantly, with noise artifacts gradually reduced.

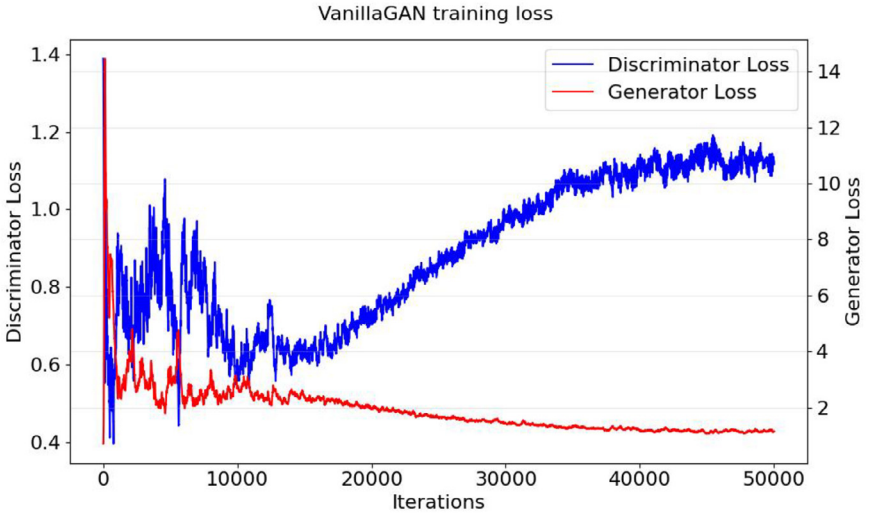
The CGAN exhibits stable convergence dynamics as shown in Fig. 3.2(a), and Fig. 3.2(b) demonstrates its capacity to synthesize digits conditioned on specific class labels.

In contrast, the WGAN exhibits a distinct convergence profile, as presented in Fig. 3.3(a). The generator loss shows more pronounced fluctuations, a behavior attributed to the regularization effect of the gradient penalty term. Nevertheless, the samples shown in Fig. 3.3(b) indicate that WGAN achieves higher image quality overall, with reduced background noise and clearer digit structure compared to its counterparts.

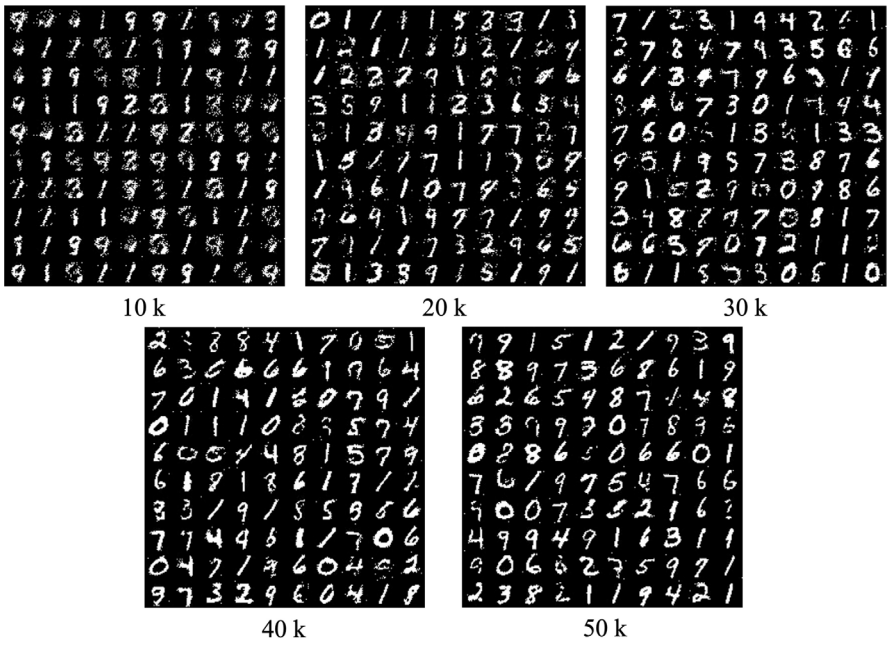
3.3.2 Style transfer and creative applications

GANs are widely recognized not only for their capacity in image generation but also for their versatility in creative tasks such as image-to-image translation and style transfer. This section presents the application of CycleGAN [23] and Pix2Pix [8] models on the Facades dataset [17], which involves translating between photographs of building facades and their corresponding architectural labels.

For CycleGAN, the generator adopts a ResNet-based architecture. It begins with reflection padding followed by a 7×7 convolution with 64 filters, Instance Normalization, and ReLU activation. The input is then downsampled twice via 3×3 convolutions, increasing the number of channels to 128 and subsequently 256. This is followed by nine residual blocks, each consisting of two 3×3 convolutions and skip connections. The network then upsamples the feature maps using two 3×3



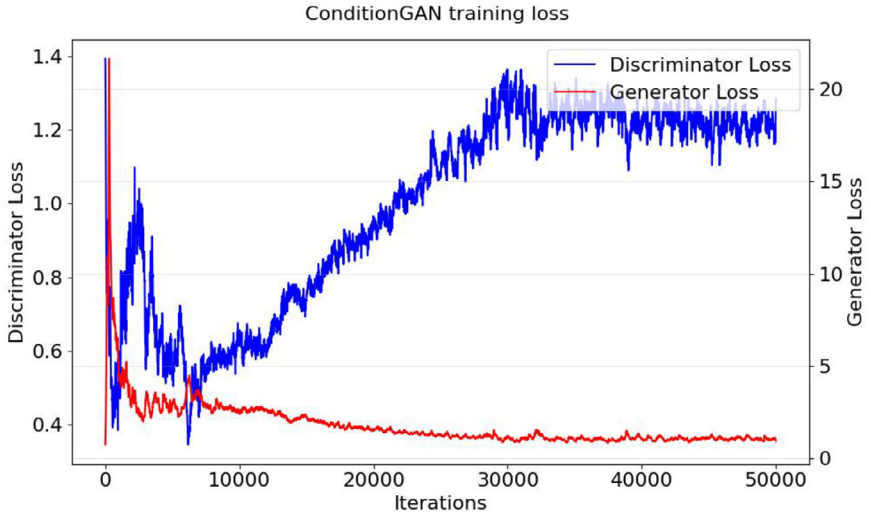
(a) Training losses of discriminator and generator v.s. iterations.



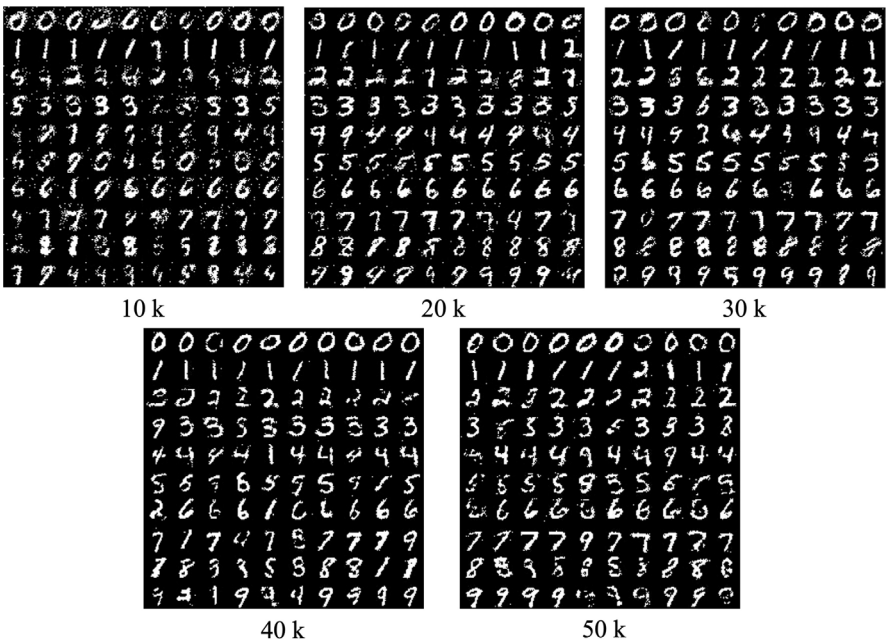
(b) Generated samples after 10k, 20k, 30k, 40k and 50k training iterations.

FIGURE 3.1

Training loss curves in 50k training iterations and sampled output images generated by Vanilla GAN after different training iterations.



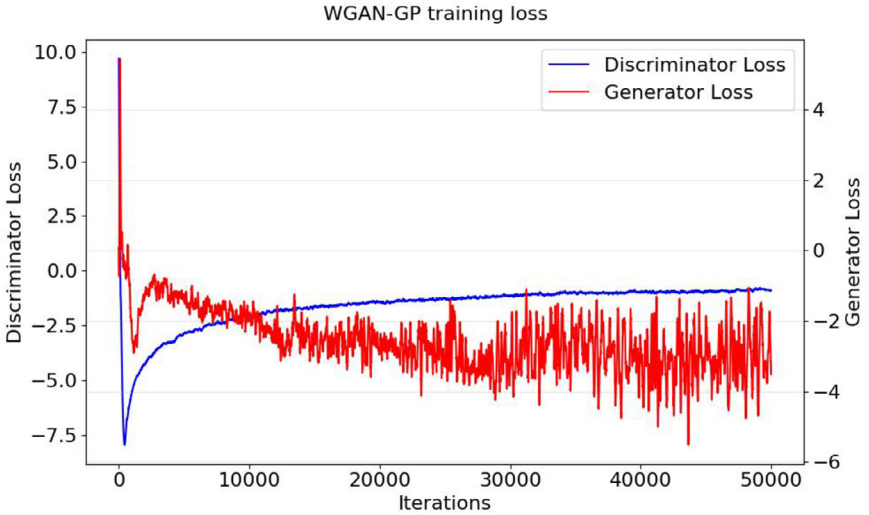
(a) Training losses of discriminator and generator v.s. iterations.



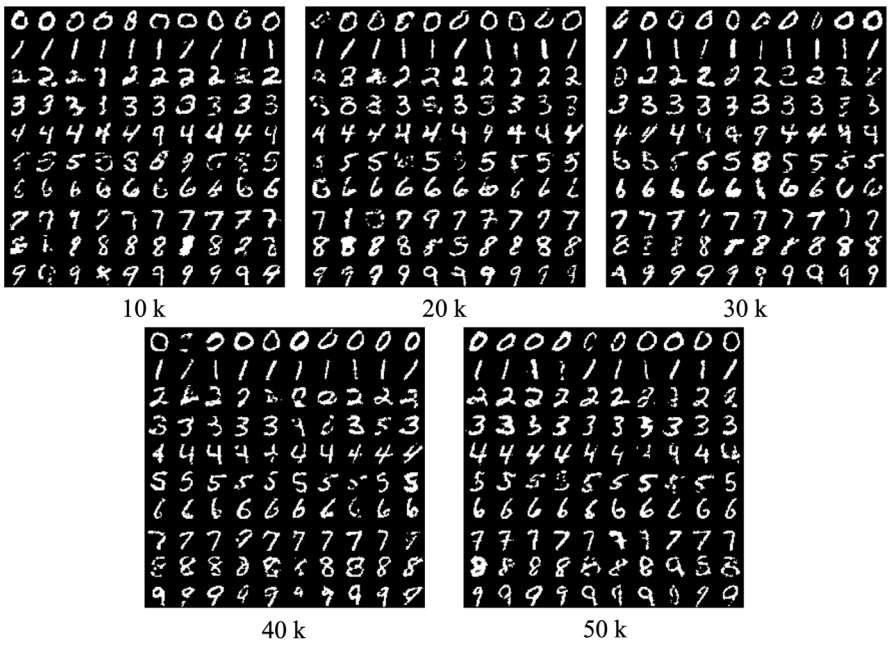
(b) Generated samples after 10k, 20k, 30k, 40k and 50k training iterations.

FIGURE 3.2

Training loss curves in 50k training iterations and sampled output images generated by CGAN after different training iterations.



(a) Training losses of discriminator and generator v.s. iterations.



(b) Generated samples after 10k, 20k, 30k, 40k and 50k training iterations.

FIGURE 3.3

Training loss curves in 50k training iterations and sampled output images generated by WGAN after different training iterations.

transposed convolutions, reducing the channel dimensions back to 128 and then 64. The final output is produced through reflection padding, a 7×7 convolution projecting to three RGB channels, and a Tanh activation to ensure pixel values fall within $[-1, 1]$.

The Pix2Pix model utilizes an encoder-decoder architecture with skip connections. The encoder compresses the input image through eight stride-2 convolutional layers, increasing the channel dimension from 3 to 512. At each stage, skip connections are established between corresponding encoder and decoder layers. The decoder upsamples the features using transposed convolutions and maps the output to RGB channels, followed by Tanh activation for normalization.

Both models share the same discriminator architecture, which concatenates the input and target images and processes them through four convolutional layers with LeakyReLU activation and batch normalization. The final output is produced by a convolutional layer that yields a scalar score, indicating the discriminator's confidence in the realism of the image.

Model training is performed using the Adam optimizer with a learning rate of 2×10^{-4}, $\beta_1 = 0.5$, $\beta_2 = 0.999$ over a total of 200 epochs. The loss functions differ between the two models. In Pix2Pix, the discriminator loss follows the standard GAN formulation, while the generator loss comprises two components: the standard adversarial loss and an L_1 reconstruction loss,

$$G_{L_1} = \mathbb{E}[\|y - G(x)\|_1],$$

which penalizes discrepancies between the generated image and the ground truth.

CycleGAN introduces a more complex loss structure, involving two generators (G_A and G_B) and two discriminators (D_A and D_B) to enable bidirectional image translation. The discriminator loss is the sum of the standard adversarial losses for both D_A and D_B. The generator loss includes the adversarial loss, a cycle consistency loss to ensure round-trip reconstruction, and an identity loss to preserve image content while transferring style. Let G_A translate domain X to Y, and G_B translate Y to X. The cycle consistency loss is defined as

$$L_{\text{cycle}} = \mathbb{E}[\|G_B(G_A(x)) - x\|_1] + \mathbb{E}[\|G_A(G_B(y)) - y\|_1],$$

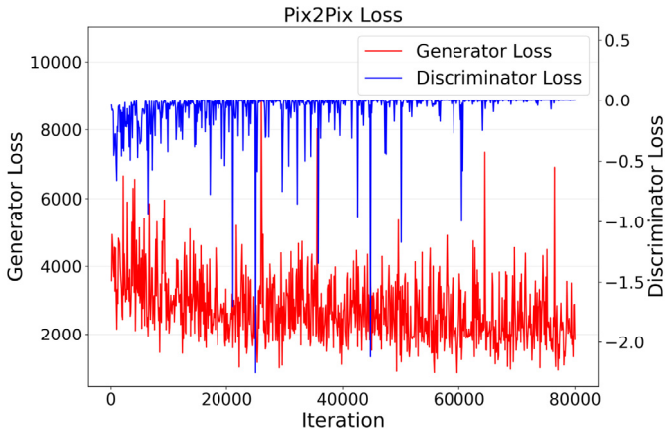
and the identity loss is given by

$$L_{\text{idt}} = \mathbb{E}[\|G_A(x) - x\|_1] + \mathbb{E}[\|G_B(y) - y\|_1].$$

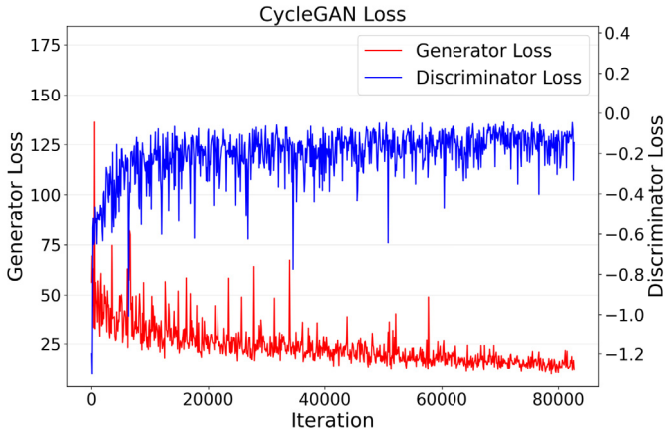
The total generator loss is expressed as

$$L_{\text{gen}} = L_{G_A} + L_{G_B} + \lambda_1 L_{\text{cycle}} + \lambda_2 L_{\text{idt}}, \tag{3.5}$$

where λ_1 and λ_2 are hyperparameters controlling the relative importance of each term.



(a) Pix2Pix Loss



(b) CycleGAN Loss

FIGURE 3.4

Training loss curves for Pix2Pix and CycleGAN.

The training loss curves for Pix2Pix and CycleGAN are shown in Fig. 3.4(a) and Fig. 3.4(b), respectively. Examples of style transfer results are presented in Fig. 3.5. The left column shows input images and their corresponding ground-truth labels from the test set. The middle two columns display the outputs generated by the trained Pix2Pix and CycleGAN models, illustrating their ability to capture the structural and stylistic features of the target domain.

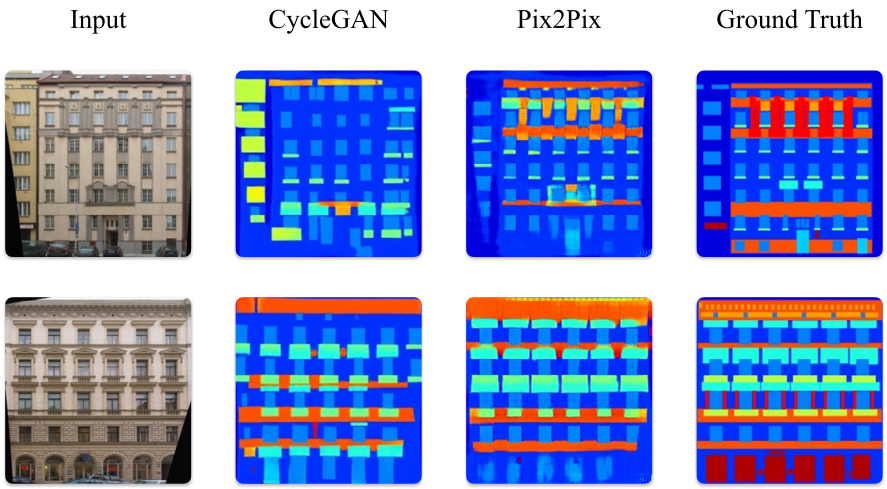


FIGURE 3.5

Sampled output images generated by different models for style transfer.

3.4 Applications of GANs in wireless communications

GANs have emerged as powerful tools in wireless communications, offering novel capabilities for addressing data scarcity, modeling complexity, and learning-based synthesis. Their generative power enables the creation of high-fidelity synthetic data, which serves as a critical asset in enhancing various wireless communication techniques.

One of the most prominent applications of GANs in this domain is wireless channel modeling. GANs can learn to generate realistic channel responses that effectively capture spatial, temporal, and environmental variations, outperforming traditional stochastic or analytical modeling approaches [21]. Furthermore, GANs have been applied to data augmentation in scenarios with limited labeled data, significantly improving the performance of supervised learning models in tasks such as automatic modulation classification [2], spectrum sensing [4], and channel estimation [5]. CGANs further extend this capability by enabling the generation of samples conditioned on specific communication attributes, such as signal-to-noise ratio levels or user locations, thereby enhancing the relevance and diversity of synthetic datasets.

Beyond synthetic data generation, GANs have also been applied to inverse problems in wireless systems. For example, they have been employed to reconstruct channel state information from compressed feedback [20]. In addition, GANs have been used to enhance the ViterbiNet framework by learning channel transition probabilities directly from receiver observations [19]. This design is particularly

advantageous in time-varying channels, where explicit modeling becomes infeasible or inaccurate.

Recent research also highlights the role of GANs in enhancing physical-layer security. Adversarial training techniques based on GANs have been proposed to improve the robustness of communication systems against jamming and spoofing attacks [22]. The unique ability of GANs to approximate complex and high-dimensional data distributions without requiring explicit analytical models makes them especially suitable for modern wireless environments characterized by high variability, uncertainty, and limited domain-specific priors.

References

[1] Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks. In: International conference on machine learning; 2017. p. 214–23.

[2] Bu K, He Y, Jing X, Han J. Adversarial transfer learning for deep learning based automatic modulation classification. IEEE Signal Processing Letters 2020;27:880–4.

[3] Daskalakis C, Panageas I. The limit points of (optimistic) gradient descent in min-max optimization. In: Advances in neural information processing systems; 2018.

[4] Davaslioglu K, Sagduyu YE. Generative adversarial learning for spectrum sensing. In: IEEE international conference on communications; 2018.

[5] Dong Y, Wang H, Yao YD. Channel estimation for one-bit multiuser massive MIMO using conditional GAN. IEEE Communications Letters 2020;25:854–8.

[6] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. Communications of the ACM 2020;63:139–44.

[7] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of Wasserstein GANs. In: Advances in neural information processing systems; 2017.

[8] Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: IEEE conference on computer vision and pattern recognition; 2017.

[9] Jin C, Netrapalli P, Jordan M. What is local optimality in nonconvex-nonconcave minimax optimization? In: International conference on machine learning; 2020. p. 4880–9.

[10] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 2002;86:2278–324.

[11] Li J, Zhu L, So AMC. Nonsmooth nonconvex–nonconcave minimax optimization: primal–dual balancing and iteration complexity analysis. Mathematical Programming 2025:1–51.

[12] Lu S. TSP: a two-sided smoothed primal-dual method for nonconvex bilevel optimization. In: International conference on machine learning; 2025.

[13] Lu S, Singh R, Chen X, Chen Y, Hong M. Alternating gradient descent ascent for nonconvex min-max problems in robust learning and GANs. In: 2019 53rd Asilomar conference on signals, systems, and computers; 2019. p. 680–4.

[14] Lu S, Tsaknakis I, Hong M. Block alternating optimization for non-convex min-max problems: algorithms and applications in signal processing and communications. In: IEEE international conference on acoustics, speech and signal processing; 2019. p. 4754–8.

[15] Lu S, Tsaknakis I, Hong M, Chen Y. Hybrid block successive approximation for one-sided non-convex min-max problems: algorithms and applications. IEEE Transactions on Signal Processing 2020;68:3676–91.
[16] Mirza M, Osindero S. Conditional generative adversarial nets. arXiv preprint. arXiv: 1411.1784, 2014.
[17] Tyleček R, Šára R. Spatial pattern templates for recognition of objects with regular structure. In: Pattern recognition, Saarbrucken, Germany; 2013. p. 364–74.
[18] Razaviyayn M, Huang T, Lu S, Nouiehed M, Sanjabi M, Hong M. Nonconvex min-max optimization: applications, challenges, and recent theoretical advances. IEEE Signal Processing Magazine 2020;37:55–66.
[19] Sun L, Wang Y, Swindlehurst AL, Tang X. Generative-adversarial-network enabled signal detection for communication systems with unknown channel models. IEEE Journal on Selected Areas in Communications 2020;39:47–60.
[20] Tolba B, Elsabrouty M, Abdu-Aguye MG, Gacanin H, Kasem HM. Massive MIMO CSI feedback based on generative adversarial network. IEEE Communications Letters 2020;24:2805–8.
[21] Yang Y, Li Y, Zhang W, Qin F, Zhu P, Wang CX. Generative-adversarial-network-based wireless channel modeling: challenges and opportunities. IEEE Communications Magazine 2019;57:22–7.
[22] Zhao C, Du H, Niyato D, Kang J, Xiong Z, Kim DI, Shen X, Letaief KB. Generative AI for secure physical layer communications: a survey. IEEE Transactions on Cognitive Communications and Networking 2024.
[23] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE international conference on computer vision; 2017.