A Formal Report on a Robust Generative Adversarial Network with Adversarial Perturbations

Debajyoti Karmaker

April 11, 2025

Abstract

This report presents a formal description of an extended Generative Adversarial Network (GAN) that incorporates an explicit adversarial attacker into the training process. The framework is designed to improve the robustness of the generator by taking into account worst-case perturbations during training. In this document, we outline the roles of each player in the three-player minimax game, derive the corresponding mathematical formulations, and explain the implementation details.

1 Introduction

Generative Adversarial Networks (GANs) typically involve a two-player game between a **Generator** and a **Discriminator**. In the proposed framework, we extend this classical setup by integrating an **Attacker** that injects adversarial perturbations into the inputs. The objective is to enforce robustness in the GAN by ensuring that the generator produces realistic outputs even when the discriminator faces worst-case perturbations.

2 Game-Theoretic Formulation

2.1 Players and Their Roles

The system is comprised of three players:

- **Generator (G):** Given a latent vector z sampled from a noise distribution $p_z(z)$, the generator produces a synthetic image:

$$\mathbf{x}_{\text{fake}} = G(z).$$

- **Discriminator (D):** The discriminator $D(\cdot)$ evaluates whether an input image $\mathbf{x}$ is real (from the data distribution p_{data}) or generated. Its output is a probability value:

$$D(\mathbf{x}) \in [0, 1].$$

- **Attacker (A):** The attacker $A(\cdot)$ is responsible for producing a perturbation δ to degrade the performance of the discriminator. The perturbation is constrained in magnitude:
$$\|A(x)\| \leq \epsilon,$$
where ϵ is a predefined threshold.

2.2 Extended Minimax Game

The classical GAN minimax game is modified to incorporate the adversarial attacker. The value function of the extended game is given by:

$$\min_{G,D} \max_{A} V(G, D, A) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D\big(x + A(x)\big) \right] + \mathbb{E}_{z \sim p_z} \left[\log \left(1 - D\big(G(z) + A\big(G(z)\big)\big)\right) \right].$$

Subject to:
$$\|A(x)\| \leq \epsilon.$$

This formulation encapsulates the idea that the generator and discriminator are optimizing their objectives under the worst-case perturbations generated by the attacker.

3 Implementation Details

3.1 Data Preprocessing and Loader

The dataset (e.g., CIFAR-10 or MNIST) is loaded and preprocessed by:

- Resizing the images to a fixed dimension (e.g., 32×32).
- Converting the images to a tensor representation.
- Normalizing pixel values using the transformation:
$$x_{\text{norm}} = \frac{x - 0.5}{0.5},$$
which scales the input pixels to the interval $[-1, 1]$.

A DataLoader is then used to efficiently provide batches of images during training.

3.2 Network Architectures

The networks for G, D, and A are implemented as follows:

- **Generator (G):** Consists of a linear layer followed by several transpose convolution layers. The architecture converts a latent vector z into an image. Nonlinear activations (ReLU) and a final tanh activation (to enforce the $[-1, 1]$ range) are used.

- **Discriminator (D):** Utilizes convolutional layers with LeakyReLU activations and batch normalization to extract features from the input image. A sigmoid function at the end produces a probability score.

- **Attacker (A):** A compact convolutional network that outputs a perturbation, which is then scaled by ϵ to ensure:
$$A(x) = \text{Net}(x) \cdot \epsilon.$$

3.3 Training Process

The training process involves an alternating optimization strategy across the three players:

1. **Attacker Update (Maximization):** With G and D fixed, the attacker updates its parameters to maximize the discriminator's loss. The loss for the attacker is given by:
$$\mathcal{L}_A = -\left(\log D\left(x + A(x)\right) + \log\left(1 - D\left(G(z) + A\left(G(z)\right)\right)\right)\right).$$

2. **Discriminator Update (Minimization):** The discriminator updates its weights to correctly classify the perturbed images:
$$\mathcal{L}_D = -\left(\log D\left(x + A(x)\right) + \log\left(1 - D\left(G(z) + A\left(G(z)\right)\right)\right)\right).$$

3. **Generator Update (Minimization):** The generator is trained to produce images that, even when perturbed, are classified as real by the discriminator:
$$\mathcal{L}_G = -\log D\left(G(z) + A\left(G(z)\right)\right).$$

Each of the above losses is optimized sequentially in an alternating training loop.

3.4 Evaluation Metrics

Two main metrics are used to track the progress:

- **Frechet Inception Distance (FID):** This metric measures the similarity between the real and generated image distributions. A lower FID indicates that the generated images are closer to the real data.

- **Attack Success Rate:** This metric evaluates how effective the attacker is by determining how often the discriminator is misled by the adversarial perturbations.

4 Convergence and Nash Equilibrium

At convergence, the system ideally reaches a Nash equilibrium denoted by the tuple (G^*, D^*, A^*), such that:

- A^* produces perturbations that maximally compromise D's performance given G^* and D^*.
- G^* generates images that remain realistic even after the worst-case perturbations.
- D^* accurately distinguishes between real and generated images despite adversarial modifications.

5 Conclusion

By introducing an explicit attacker into the GAN framework, the proposed model explicitly accounts for worst-case adversarial perturbations during training. The formal objective of the system is given by:

$$\min_{G,D} \max_{A} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D(x + A(x)) + \mathbb{E}_{z \sim p_z} \log\big(1 - D(G(z) + A(G(z)))\big) \right],$$

subject to the constraint $\|A(x)\| \leq \epsilon$. The alternating training strategy and the evaluation via FID and attack success rate collectively contribute to achieving a robust and secure generative model. This approach lays a theoretically sound foundation for future research in enhancing the robustness of generative networks against adversarial attacks.