Notes on Generative Models: Adversarial Autoencoders and GANs

October 9, 2025

1 Adversarial Autoencoders (AAEs)

1.1 Overview

Adversarial Autoencoders (AAEs), introduced by Makhzani et al. (ICLR 2016), combine autoencoders with adversarial training to learn meaningful latent representations and generate data samples. They enforce a prior distribution on the latent space using a discriminator, inspired by Generative Adversarial Networks (GANs).

1.2 Core Components

- Encoder: Maps input x to latent code z, i.e., $q(z|x)$.
- Decoder: Reconstructs input from z, i.e., $p(x|z)$.
- Discriminator: Ensures the latent distribution $q(z)$ matches a chosen prior $p(z)$ (e.g., Gaussian).

1.3 Training Objective

The AAE optimizes two losses:

- Reconstruction Loss: Minimizes $\|x - \hat{x}\|^2$ to ensure accurate input reconstruction.
- Adversarial Loss: Matches $q(z)$ to $p(z)$ using a GAN-like objective:

$$\min_q \max_D \mathbb{E}_{p(z)}[\log D(z)] + \mathbb{E}_{q(z)}[\log(1 - D(z))]$$

1.4 Advantages

- Structured latent space suitable for generative tasks.
- Flexibility to use any prior distribution (e.g., Gaussian, mixture models).
- Applications in data generation, semi-supervised learning, and representation learning.

2 Adversarial Autoencoders Paper (Makhzani et al., ICLR 2016)

2.1 Key Contributions

- Introduced AAEs as a probabilistic framework combining autoencoders and adversarial training.
- Replaced the KL-divergence penalty of VAEs with adversarial training for flexible prior matching.
- Demonstrated applications in generative modeling, semi-supervised learning, and representation learning.

2.2 Methodology

- Architecture: Encoder ($q(z|x)$), decoder ($p(x|z)$), and discriminator to match $q(z)$ to $p(z)$.
- Training: Alternates between minimizing reconstruction loss and adversarial loss.
- Variants: Supervised AAEs (incorporate labels), semi-supervised AAEs, and denoising AAEs.

2.3 Experiments

- Datasets: MNIST, CIFAR-10, SVHN.
- Results: Generated sharp samples, achieved competitive semi-supervised classification, and learned meaningful latent representations.
- Comparisons: Outperformed VAEs in sample quality and flexibility, retained reconstruction unlike GANs.

2.4 Limitations

- Training instability due to adversarial component.
- Potential mode collapse in latent space.
- Higher computational cost than standard autoencoders.

3 VAEs vs. AAEs: Flexibility in Prior Matching

3.1 Statement

VAEs use KL-divergence to match the latent distribution $q(z|x)$ to a simple prior (e.g., $\mathcal{N}(0, I)$), which is restrictive for complex distributions. AAEs use adversarial training, allowing any prior distribution.

3.2 Explanation

- VAEs: Optimize:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}}(x, \hat{x}) + \text{KL}(q(z|x)||p(z))$$

KL-divergence is tractable only for simple priors (e.g., Gaussian), limiting latent space complexity.

- AAEs: Use a discriminator to match $q(z)$ to any $p(z)$, enabling complex priors like mixture models without computing KL-divergence.

3.3 Example: MNIST Digits

- VAE: Forces all digit latent codes into a single Gaussian, leading to blurry samples and poor separation of digit classes.
- AAE: Uses a mixture of Gaussians (one per digit class), producing sharper samples and a structured latent space with distinct clusters.

4 Generative Adversarial Networks (GANs)

4.1 Overview

GANs, introduced by Goodfellow et al. (2014), are generative models that train a generator and discriminator in a minimax game to produce realistic data samples.

4.2 Core Components

- Generator (G): Maps noise $z \sim p_z(z)$ (e.g., $\mathcal{N}(0, I)$) to fake samples $G(z)$.
- Discriminator (D): Outputs probability $D(x) \in [0, 1]$ to distinguish real data ($x \sim p_{\text{data}}$) from fake ($G(z)$).

4.3 Objective Function

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

At equilibrium, $p_g(x) = p_{\text{data}}(x)$, and $D(x) = 0.5$.

4.4 Training Process

- Update D to maximize discrimination accuracy.
- Update G to minimize $D(G(z))$, using non-saturating loss: $-\mathbb{E}_{z \sim p_z}[\log D(G(z))]$.

4.5 Variants

- DCGANs: Use convolutional layers for image tasks.
- Conditional GANs: Condition on labels or text.
- WGANs: Use Wasserstein distance for stability.
- CycleGAN: Unpaired image-to-image translation.
- StyleGAN: High-resolution image synthesis with style control.

4.6 Applications

- Image synthesis (e.g., faces, artwork).
- Image-to-image translation (e.g., sketches to photos).
- Data augmentation, text-to-image synthesis, audio/video generation.

4.7 Challenges

- Training instability and mode collapse.
- Difficulty in evaluating sample quality.
- High computational cost and ethical concerns (e.g., deepfakes).

4.8 Connection to AAEs

- AAEs use adversarial training to match latent $q(z)$ to $p(z)$, while GANs match generated $p_g(x)$ to $p_{\text{data}}(x)$.
- AAEs combine reconstruction (autoencoder) with generation, unlike GANs, which focus only on generation.

4.9 Example: MNIST Digits

- Generator: Maps 100D noise to 28x28 images using dense layers.
- Discriminator: Classifies 28x28 images as real or fake using CNNs.
- Outcome: Generates realistic digits, but lacks reconstruction ability unlike AAEs.