WASSERSTEIN AUTOENCODERS: A COMPARISON WITH VARIATIONAL AUTOENCODERS AND WASSERSTEIN GANS

Yuntao Wu
Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada
UTORid: wuyuntao
winstonyt.wu@mail.utoronto.ca

1 Introduction

Consider a dataset $X = \{x_i\}_{i=1}^N$, consisting of N i.i.d. samples of images $x_i \in \mathbb{R}^{H\times W\times C}$, where H is the height, W is the width, and C is the number of channel of an image. If an image is of grey scale, then $C = 1$, while for an RGB image, $C = 3$. These samples are drawn from some unknown probability distribution μ on $\mathbb{R}^{H\times W\times C}$ s.t. $\mu(X) = 1$. The goal is to generate synthetic images y_i that resemble real images from X. To achieve this, we define a latent space Z, from which we sample random noise according to a known prior distribution η, typically a Gaussian distribution, *i.e.* $\eta = \mathcal{N}(0, I)$, where I is the identity matrix. Since the Gaussian measure η is absolutely continuous with respect to the Lebesgue measure, it admits a probability density function $p(z)$ such that $p(z) = \frac{d\eta}{dz}$. Consequently, the total probability satisfies $\int_Z p(z)dz = \int_Z d\eta(z) = 1$. Let $G_\theta : Z \to \mathbb{R}^{H\times W\times C}$ be a generator, parameterized by θ, that maps a latent variable $z \sim \eta$, to a synthetic image $y = G_\theta(z) \in \mathbb{R}^{H\times W\times C}$. The generator G_θ introduces a probability distribution $(G_\theta)_\#\eta$, which is the push-forward probability measure of η under G_θ. To generate realistic images, we aim to find parameters θ such that the generated distribution $(G_\theta)_\#\eta$ approximates the true data distribution μ. In other words, if we denote by p_θ the probability distribution of generated images, we seek to maximize the likelihood of real samples under this distribution. The marginal likelihood of an image x is given by:

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz.$$

Since X consists of i.i.d. samples, the optimal parameter θ^* is obtained by maximizing the joint likelihood:

$$\theta^* = \arg\max_\theta p_\theta(X) = \arg\max_\theta \prod_{i=1}^N p_\theta(x_i).$$

However, direct optimization of this objective is challenging, because for small values of $p_\theta(x_i)$, the product can become extremely small, leading to numerical instability. To mitigate this, we instead maximize the log-likelihood:

$$\theta^* = \arg\max_\theta \log p_\theta(X) = \arg\max_\theta \log \prod_{i=1}^N p_\theta(x_i) = \arg\max_\theta \sum_{i=1}^N \log p_\theta(x_i).$$

Unfortunately, the marginal likelihood $p_\theta(x)$ is generally intractable, preventing direct evaluation and differentiation. Additionally, the true posterior distribution $p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$ is also intractable, making it difficult to apply expectation-maximization algorithms [9]. To address these challenges, several methods have been developed, including variational autoencoders (VAEs), generative adversarial networks (GANs) and diffusion models, each with various refinements [9, 7, 4, 12, 2, 15, 5, 8, 14].

In this project, we focus on a specific variant of VAEs known as Wasserstein Autoencoders (WAEs) [17]. WAEs combine the latent-variable modeling approach of VAEs with optimal transport theory, aiming to generate realistic images like GANs while maintaining the training stability of VAEs. We compare their performance against β-VAEs and WGAN-GP [7, 5] on the MNIST and CelebA datasets. Specifically, we seek to answer the following questions:

1. (RQ1) What are the key similaries and differences among VAE, WAE, WGAN-GP?
2. (RQ2) How efficiently can β-VAE and WAE encode images to latent space (Gaussian noise)?
3. (RQ3) How well do VAEs, WAEs and WGAN-GP generate images?

2 Background

VAEs are based on the idea of variational inference, while WAEs and WGAN-GP are based on optimal transport theory. In this section, some theoretical backgrounds of the two topics are provided.

2.1 Variational Inference

The main idea of variational method is to transform inference into an optimaization problem. We aim to approximate the marginal likelihood of an image $p_\theta(x)$, which is generally intractable. Instead, we approximate the posterior distribution $p(z|x)$ over a set of unobserved latent variables Z given input data X with some tractable distribution $q(z|x)$, such as a Gaussian. This approximation is called the variational distribution. To measure the similarity between two probability distributions $p(x)$ and $q(x)$, a dissimilarity function $D(p||q)$ is used. Inference is performed by selecting the distribution $q(z|x)$ that minimizes the divergence $D(q(z|x)||p_\theta(z))$. The most common type of variational Bayes method uses the Kullback-Leibler divergence (KL-divergence) as the dissimilarity function:

$$D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx.$$

Note that $D_{KL}(p||q) \geq 0$ for any probability distributions p and q, and $D_{KL}(p||q) = 0$ if and only if $p = q$. However, $D_{KL}(p||q) \neq D_{KL}(q||p)$, and $D_{KL}(p||q) \not\leq D_{KL}(p||r) + D_{KL}(r||q)$, so it is not symmetric and does not satisfy the triangle inequality; therefore, it is not a proper distance metric.

Consider the image space X with probability distribution $p_\theta(x)$ and latent space Z with prior probability distribution $p(z)$, as defined in Section 1. Variational inference indirectly optimizes the posterior approximation:

$$\begin{aligned}
\log p_\theta(x) &= \int \log p_\theta(x) p(z) dz = \int \log \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} p(z) dz \\
&= \int \log p_\theta(x|z) p(z) dz + \int p(z) \log \frac{p(z)}{p_\theta(z|x)} dz - \int p(z) \log \frac{p(z)}{p_\theta(z)} dz \\
&= \mathbb{E}_z[p_\theta(x|z)] + D_{KL}(p(z)||p_\theta(z|x)) - D_{KL}(p(z)||p_\theta(z)).
\end{aligned}$$

By instead maximizing the evidence lower bound (ELBO):

$$\text{ELBO}(p) = \mathbb{E}_z[p_\theta(x|z)] - D_{KL}(p(z)||p_\theta(z)) \leq \log p_\theta(x),$$

we obtain a tractable optimization objective. Note that $p(z)$ is a known probability distribution, $p_\theta(x|z)$ and $p_\theta(z)$ can be computed by proper reparametrization as discussed in Section 3.1.

2.2 Optimal Transport

Another way to formulate the generation problem is to define a proper distance between the true probability distribution $p(x)$ and the approximated probability distribution $p_\theta(x)$ and minimize this distance. We follow the notations in [13]. Let X, Y be complete separable metric spaces bounded in $\mathbb{R}^{H \times W \times C}$, where X represents real images, and Y represents generated images. Let μ, ν be probability measures on $\mathbb{R}^{H \times W \times C}$ s.t. $\mu(X) = \nu(Y) = 1$. Let $c : X \times Y \to \mathbb{R}$ s.t. $(x, y) \mapsto c(x, y)$ be a function, representing the cost to transport from x to y. Following Kantorovich's (1942) formulation, we seek a probability measure γ on $X \times Y$, with marginals μ and ν that minimizes the total transport cost:

$$\inf_{\gamma \in \Pi(\mu, \nu)} \int_{X \times Y} c(x, y) d\gamma.$$

Here, $\Pi(\mu, \nu)$ represents the set of admissible transport plans γ such that $(\Pi_X)_\# \gamma = \mu$ and $(\Pi_Y)_\# \gamma = \nu$, where $\Pi_X(x, y) = x$, and $\Pi_Y(x, y) = y$ are projection maps. Assume that $\mu, \nu \ll \mathcal{L}$ (Lebesgue measure on $\mathbb{R}^{H \times W \times C}$) and $\gamma \ll \mathcal{L}$ on $\mathbb{R}^{H \times W \times C} \times \mathbb{R}^{H \times W \times C}$ so that the Radon-Nikodym derivatives $p(x) = \frac{d\mu}{dx}$, $p_\theta(y) = \frac{d\nu}{dy}$ and $\gamma(x, y) = \frac{d\gamma}{dxdy}$ are well-defined.

The Kantorovich problem is a linear programming problem and has a dual form:

$$\sup_{\phi, \psi \in C_b} \left\{ \int_X \phi(x) d\mu(x) + \int_Y \psi(y) d\nu(y) : \phi(x) + \psi(y) \leq c(x, y) \right\}$$

Define the c-transform of $\phi(x)$ as

$$\phi^c(y) = \inf_x c(x, y) - \phi(x).$$

If X, Y are compact metric spaces with $c(x, y) = \|x - y\|_1$ as a continuous distance metric on $X \times Y$, then there exists a bounded Lipschitz-1 function u (Kantorovich potential) such that $|u(x) - u(y)| \leq \|x - y\|_1$, solving the dual Kantorovich problem:

$$\sup_{u \in \text{Lip-1}} \int_X u d\mu - \int_Y u d\nu$$

Now, consider the cost function $c(x, y) = \|x - y\|_p^p$, which represents the p-th power of the L^p-distance for $p \geq 1$. In this case, the p-th root of the Kantorovich problem defines the Wasserstein distance $W_p(\mu, \nu)$:

$$W_p^p(\mu, \nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{X \times Y} \|x - y\|_p^p \, d\gamma$$

3 Model Architectures and Evaluation Metric

3.1 Variational Autoencoders

Variational Autoencoders (VAEs), introduced by Kingma and Welling [9], are a class of unsupervised generative models. They employ evidence lower bound (ELBO) maximization, as discussed in Section 2.1, ensuring stable training. A VAE consists of two main components: an encoder q_ϕ and a decoder p_θ, as illustrated in Figure 1a.

The encoder maps observed data $x \in X$ to a latent variable $z \in Z$ by learning the approximate posterior distribution $q_\phi(z|x)$, parameterized by ϕ. $q_\phi(z|x)$ is chosen to match a multivariate normal distribution, $p(z) = \mathcal{N}(0, 1)$. The decoder $p_\theta(z)$, parametrized by θ, reconstructs the observed data x from the latent variable z. The objective function of a VAE model is derived from the ELBO:

$$\begin{aligned} &\sup_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - D_{KL}(q_\phi(z|x)||p(z)) \\ =&\sup_{\theta,\phi} \int \log p_\theta(x|z) q_\phi(z|x) dz - \int q_\phi(z|x) \log \frac{q_\phi(z|x)}{p(z)} dz \end{aligned}$$

To control the trade-off between reconstruction accuracy and latent space regularization, Higgins et al. [7] proposed a scaling factor on the KL divergence:

$$\sup_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \beta D_{KL}(q_\phi(z|x)||p(z))$$

Changing β changes the focus of training. A small value for β means we focus on matching the probability distribution of synthetic images with the probability distribution of original images, while a larger β means we focus on matching the encoded distribution with the prior distribution.

To ensure a tractable and differentiable KL divergence term $D_{KL}(q_\phi(z|x)||p(z))$, the reparametrization trick is introduced. Instead of encoding a deterministic latent variable z, the encoder outputs two parameters μ_z and σ_z, and samples z with:

$$z = \mu_z + \sigma_z \epsilon, \epsilon \sim \mathcal{N}(0, I).$$

Maximizing the expectation $\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right]$ is equivalent to minimizing the mean squared error between original image x and the reconstructed image $\hat{x}$. The final objective function is:

$$\min L_{VAE}(x, \hat{x}) = \min \frac{1}{2} \|x - \hat{x}\|_{L^2}^2 - \frac{\beta}{2} \sum_i (1 + \log \sigma_i^2 - \sigma_i^2 - \mu_i^2),$$

where $i \in [n]$ indexes the latent variables $z = (z_1, z_2, ..., z_n)$.

With a single objective function, we can optimize both the encoder q_ϕ and decoder p_θ simultaneously using Algorithm 1.

Since the latent space dimension n is typically much smaller than the input dimension, VAEs function as lossy compressors, extracting the top principal components of the data. Consequently, reconstructed images tend to be blurry, as high-frequency details are lost during compression.

3.2 Wasserstein GANs

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [4], estimate the probability distribution $p(x)$ of real images through an adversarial training process. As illustrated in Figure 1b, a GAN consists of two competing

Algorithm 1 VAE Training

Input: $X = \{x_i\}_{i=1}^N$: dataset consisting of N i.i.d. samples of images $x_i \in \mathbb{R}^{H \times W \times C}$
q_ϕ: Encoder with parameters ϕ
p_θ: Decoder with parameters θ
m: batch size
α, β_1, β_2: learning rate and weights for running averages

while (ϕ, θ) not converged **do**
  Sample a batch $\{x_1, x_2, ..., x_m\}$ from the training set.
  $L \leftarrow 0$
  for $i = 1$ **to** m **do**
    $\mu_{z,i}, \sigma_{z,i} \leftarrow q_\phi(x_i)$
    $\epsilon \sim \mathcal{N}(0, I)$
    $z_i \leftarrow \mu_{z,i} + \sigma_{z,i}\epsilon$
    $\hat{x}_i \leftarrow p_\theta(z_i)$
    $L \leftarrow L + L_{VAE}(x_i, \hat{x}_i)$
  end for
  Update (ϕ, θ) with Adam($\frac{1}{m}L$, α, β_1, β_2)
end while

models: a generator G_θ, which learns to capture the data distribution, and a discriminator D_ϕ, which distinguishes between real and generated samples. This setup forms a two-player minimax game with value function $V(D_\phi, G_\theta)$, aiming to minimize the lower bound on the Jensen-Shannon divergence[1]:

$$\min_{G_\theta} \max_{D_\phi} V(D_\phi, G_\theta) = \min_{G_\theta} \max_{D_\phi} \int \log D_\phi(x) d\mu + \int \log(1 - D_\phi(G_\theta(z))) d\eta,$$

where μ represents the probability measure of real images, and η is a prior distribution like Gaussian, as defined in Section 1.

However, GAN training can be unstable, particularly when the distributions of real and generated images have disjoint support, making the Jensen-Shannon divergence ineffective. To address this, Wasserstein GANs (WGANs) [2] reformulate the loss function as an optimal transport problem, minimizing the Wasserstein distance $W_p(\mu, \nu)$ between the real data distribution μ, and the generated distribution ν. Training in WGANs involves two main steps. First, the critic (the discriminator) is trained to approximate the Kantorovich potential by solving the dual Kantorovich problem:

$$\max_{D_\phi} L_{WGAN,D}(x, z) = \max_{D_\phi} \int_X D_\phi(x) d\mu - \int_Z D_\phi(G_\theta(z)) d\eta$$

Next, the generator is updated by minimizing the gradient of the Wasserstein-1 distance:

$$\min_{G_\theta} L_{WGAN,G}(z) = \min_{G_\theta} \nabla_\theta W_1(\mu, \nu) = \min_{G_\theta} \int -\nabla_\theta D_\phi(G_\theta(z)) d\eta$$

To stabilize training, the critic is typically updated for $n_{\text{critic}} > 1$ iterations before each generator update.

Despite this improvement, the original WGAN formulation does not fully enforce the Lipschitz-1 constraint on D_ϕ, as weight clipping alone does not guarantee a valid Lipschitz function [15]. A more effective approach is to introduce a gradient penalty when solving the dual Kantorovich problem, ensuring that the Lipschitz constraint is met [5, 11]. This is achieved by penalizing deviations in the gradient norm: $\|\nabla_x D_\phi(x)\|_2$. However, it is impractical to compute this gradient for all real and generated samples, so instead, the gradient is evaluated at points $\tilde{x}$, sampled uniformly along straight lines between pairs of points from real and generated samples. The revised objective function for WGAN-GP (Wasserstein GAN with Gradient Penalty) are:

$$\min_{D_\phi} L_{WGAN-GP,D}(x, z) = \min_{D_\phi} \int_Z D_\phi(G_\theta(z)) d\eta - \int_X D_\phi(x) d\mu + \lambda \int \left(\|\nabla_{\tilde{x}} D_\phi(\tilde{x})\|_2 - 1\right)_+^2 \sigma(\tilde{x}) d\tilde{x},$$

$$\min_{G_\theta} L_{WGAN-GP,G}(z) = \min_{G_\theta} \int -\nabla_\theta D_\phi(G_\theta(z)) d\eta,$$

where λ is a regularization factor applied to the penalty term, typically set to 10, and $(x)_+ = \max(x, 0)$. We use $(x)_+$ so that the penalty term is applied only when $\|\nabla_{\tilde{x}} D_\phi(\tilde{x})\|_2 > 1$, guiding the critic to satisfy the Lipschitz constant. The training procedure is outlined in Algorithm 2.

[1] $V(D_\phi, G_\theta) \leq 2D_{JS}(\mu, \nu) - \log(4)$ as the lower bound for the Jensen-Shannon divergence over real and generated distribution [17].

Algorithm 2 WGAN-GP Training

Input: $X = \{x_i\}_{i=1}^{N}$: dataset consisting of N i.i.d. samples of images $x_i \in \mathbb{R}^{H \times W \times C}$
D_ϕ: critic/discriminator with parameters ϕ
G_θ: generator with parameters θ
λ: gradient penalty coefficient
m: batch size
$n_{\text{critic}} = 5$: number of critic iterations per generator iteration
α, β_1, β_2: learning rate and weights for running averages

while θ not converged **do**
  for $t = 1$ **to** n_{critic} **do**
    Sample a batch $\{x_1, x_2, ..., x_m\}$ from the training set.
    Sample a batch of latent variables $\{z_1, z_2, ..., z_m\}$ from $\mathcal{N}(0, I)$.
    $L \leftarrow 0$
    for $i = 1$ **to** m **do**
      Sample interpolation constant $\epsilon \sim$ Unif$[0, 1]$.
      $\hat{x}_i \leftarrow G_\theta(z_i)$
      $\tilde{x}_i \leftarrow \epsilon x_i + (1 - \epsilon)\hat{x}_i$
      $L \leftarrow L + L_{WGAN-GP,D}(x_i, z_i)$
    end for
    Update ϕ with Adam($\frac{1}{m}L$, α, β_1, β_2)
  end for
  Update θ with Adam($\frac{1}{m}\sum_{i=1}^{m} -D_\phi(G_\theta(z_i))$, α, β_1, β_2)
end while

3.3 Wasserstein Autoencoders

Variational Autoencoders (VAEs) are theoretically well-established and offer stable training, but they often produce blurry samples. In contrast, Generative Adversarial Networks (GANs) generate more realistic images but are harder to train due to the absence of an encoder and may fail to capture all the variability in the true data distribution. To address these limitations, Wasserstein Autoencoders (WAEs) were introduced [17]. As illustrated in Figure 1c, a WAE integrates concepts from both VAEs and GANs, comprising three key components: an encoder $q_\phi(z|x)$, a decoder $p_\theta(x|z)$, and a discriminator $D_\psi(z)$. The goal is to minimize the transport cost between the data distribution (X, μ) and the generated distribution (Y, ν), which corresponds to solving the primal Kantorovich problem:

$$\inf_{\gamma \in \Pi(\mu,\nu)} \int_{X \times Y} c(X, Y) d\gamma$$

However, directly computing this value is intractable, as the data distribution is unknown, and the generated distribution is parameterized by neural networks. Instead of solving for γ explicitly, it suffices to find a conditional distribution $q(z|x)$ such that its Z marginal $q_(z) = \int_X q(z|x)d\mu$ matches the prior distribution $p(z)$. This transformation simplifies the primal Kantorovich problem, leading to the following Theorem 1:

Theorem 1. *Let ν be the generated distribution with a deterministic generator $p_\theta : Z \to X$. Then*

$$\inf_{\gamma \in \Pi(\mu,\nu)} \int_{X \times Y} c(x, y) d\gamma = \inf_{q: q(z) = p(z)} \int_X \int_Z c(x, p_\theta(z)) q(z|x) dz d\mu,$$

where $q(z) = \int_X q(z|x)d\mu$ is the marginal distribution.

Proof. Let (X, μ) denote the space of real images, (Y, ν) the space of generated images, and (Z, η) the space of latent variables. Assume their respective measures are absolutely continuous with respect to Lebesgue measure, with density functions:

$$p_X(x) = \frac{d\mu}{dx}, \quad p_Y(y) = \frac{d\nu}{dy}, \quad p_Z(z) = \frac{d\eta}{dz}, \quad p_{XY}(x, y) = \frac{d\gamma}{dxdy}$$

Define $\mathcal{P}(X, Z)$ and $\mathcal{P}(X, Y, Z)$ as the sets of joint probability distributions over (X, Z) and (X, Y, Z), respectively.

Since the generator deterministically maps $z \in Z$ to $\hat{x} \in Y$, Y is independent of X conditioned on Z.

$$(Y \perp X)|Z, p(x, y|z) = p(x|z)p(y|z)$$

Thus, we can rewrite the primal Kantorovich problem:

$$\inf_{\gamma \in \Pi(\mu,\nu)} \int_{X\times Y} c(x,y)d\gamma(x,y) = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{X\times Y} c(x,y)p_{XY}(x,y)dxdy$$

Applying conditional independence:

$$= \inf_{p\in\mathcal{P}(X,Y,Z)} \int_Z \int_{X\times Y} c(x,y)p(x,y|z)p_Z(z)dxdydz$$

$$= \inf_{p\in\mathcal{P}(X,Y,Z)} \int_Z \int_X \int_Y c(x,y)p(x|z)p(y|z)p_Z(z)dydxdz$$

Substituting $y = p_\theta(z)$:

$$= \inf_{p\in\mathcal{P}(X,Y,Z)} \int_Z \int_X c(x,p_\theta(z))p(x|z)p_Z(z)dxdz$$

$$= \inf_{p\in\mathcal{P}(X,Z)} \int_Z \int_X c(x,p_\theta(z))p(x|z)p_Z(z)dxdz$$

Using Bayes' rule on X, Z :

$$= \inf_q \int_X \int_Z c(x,p_\theta(z))q(z|x)dz(p_X(x)dx)$$

$$= \inf_q \int_X \int_Z c(x,p_\theta(z))q(z|x)dzd\mu$$

This completes the proof. □

Theorem 1 enables optimization over encoders rather than directly computing transport costs between real and generated data distributions. To relax the functional constraint $q(z) = p(z)$, a penalty term is introduced via the discriminator $D_\psi(z)$, leading to the following objective function:

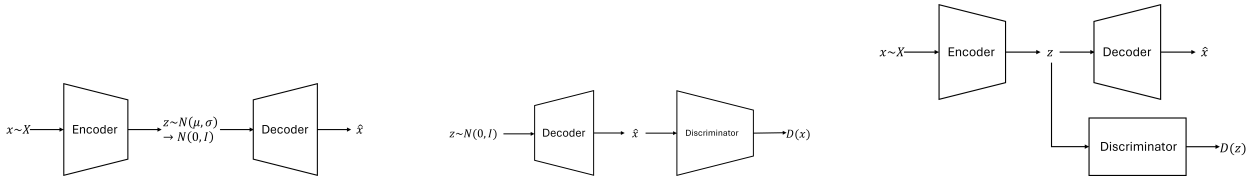
$$\min_{\theta,\phi} L_{WAE}(x,z) = \min_{\theta,\phi} \int_X \int_Z c(x,p_\theta(z))q_\phi(z|x)dzd\mu + \lambda D_Z(q_\phi(z|x),p(z)),$$

where $D_Z(q_\phi(z|x),p(z))$ represents a divergence measure between $q_\phi(z|x)$ and $p(z)$. The authors of [17] propose two penalty approaches: one based on adversarial training (GAN-based) and another using maximum mean discrepancy (MMD-based). We consider the GAN-based approach:

$$\sup_\psi \int_Z \log D_\psi(z)d\eta + \int_X \log(1 - D_\psi(q_\phi(z|x)))d\mu.$$

Although this results in a min-max problem similar to GANs, the discriminator operates in latent space rather than data space, where the probability distribution is known. The training procedure is outlined in Algorithm 3, with the cost function defined as $c(x,y) = \|x-y\|_2^2$, minimizing the Wasserstein-2 distance. This approach aligns with adversarial auto-encoders (AAEs) proposed by [10].

Unlike VAEs, which use a probabilistic encoder and require the reparameterization trick, WAEs allow for non-random encoders that deterministically map inputs to their latent codes. The decoder, similar to both the decoder in VAEs and generator in GANs, takes a latent code z and generates a synthetic image. Instead of discriminating between real and generated images, as in standard GANs, or computing the Kantorovich potential, as in WGANs, the discriminator D_ϕ in WAEs minimizes the discrepency between mapped latent distribution $q_\phi(z|x)$ and the true prior distribution $p(z)$.



(a) VAE (b) WGAN-GP (c) WAE

Figure 1: Architecture

Algorithm 3 WAE Training

Input: $X = \{x_i\}_{i=1}^{N}$: dataset consisting of N i.i.d. samples of images $x_i \in \mathbb{R}^{H \times W \times C}$
q_ϕ: encoder with parameters ϕ
p_θ: decoder with parameters θ
D_ψ: discriminator with parameters ψ
λ: regularization coefficient
m: batch size
α_1, α_2, β_1, β_2: learning rates and weights for running averages

while (ϕ, θ) not converged **do**
  Sample a batch $\{x_1, x_2, ..., x_m\}$ from the training set.
  Sample a batch of latent variables $\{z_1, z_2, ..., z_m\}$ from $\mathcal{N}(0, I)$.
  $L_D \leftarrow 0$
  $L_{WAE} \leftarrow 0$
  for $i = 1$ **to** m **do**
    $\tilde{z}_i \leftarrow q_\phi(x_i)$
    $L_D \leftarrow L_D + (\log D_\psi(z_i) + \log(1 - D_\psi(\tilde{z}_i)))$
    $L_{WAE} \leftarrow L_{WAE} + c(x_i, p_\theta(\tilde{z}_i)) - \lambda \log D_\psi(\tilde{z}_i)$
  end for
  Update ψ with Adam($-\frac{\lambda}{m} L_D$, α_1, β_1, β_2)
  Update (ϕ, θ) with Adam($\frac{1}{m} L_{WAE}$, α_2, β_1, β_2)
end while

3.4 Evaluation Metrics

To quantitatively address RQ2 and RQ3 from Section 1, we define appropriate evaluation metrics in addition to the objective functions of each model.

KL-Divergence: For RQ2, we evaluate how well β-VAE and WAE map images to latent codes following a Gaussian distribution. We use the Kullback-Leibler (KL) divergence, $D_{KL}(p||q)$, as defined in Section 3.1, which quantifies the difference between a distribution q and the true probability distribution p. However, unlike VAEs, which output both the mean and variance through the reparameterization trick, WAEs deterministically map images $x \in X$ to latent code $z \in Z$. To approximate the KL divergence, we use the squared Euclidean distance from the prior:

$$D_{KL}(q_\phi(z|x)||p(z)) \approx \frac{1}{2} \|z\|^2 .$$

This approximation holds because if $z \sim \mathcal{N}(0, 1)$, its squared norm follows a chi-squared distribution, which aligns with the KL divergence. A lower KL divergence means the latent distribution is closer to the Gaussian distribution.

Frechet Inception Distance: For RQ3, we assess how closely the generated image distribution matches the real data distribution using the Frechet Inception Distance (FID) [6], which is essentially the Wasserstein-2 distance. Given μ, ν be two probability distributions with finite mean and variance, the FID is defined as:

$$d_F(\mu, \nu) = W_2(\mu, \nu) = \left(\inf_{\gamma \in \Pi(\mu, \nu)} \|x - y\|_2^2 \, d\gamma \right)^{\frac{1}{2}}$$

For two multidimensional normal distributions, $\mu = \mathcal{N}(\mu_1, \Sigma_1)$ and $\nu = \mathcal{N}(\mu_2, \Sigma_2)$, the FID score has a closed form:

$$d_F^2(\mu, \nu) = \|\mu_1 - \mu_2\|_2^2 + \text{Tr}\left(\Sigma_1 + \Sigma_2 - 2(\Sigma_1 \Sigma_2)^{\frac{1}{2}} \right).$$

Intuitively, a lower FID score indicates that the generated image distribution is more similar to the real data distribution. Since computing FID directly on images in $\mathbb{R}^{H \times W \times C}$ is computationally expensive, FID instead compares the mean and covariance of features extracted from the final layer of Inception V3 [16], which capture the most distinguishing features of the images. As Inception V3 is pre-trained on images of size $(H, W, C) = (299, 299, 3)$, input images are interpolated to match this resolution before feature extraction.

4 Experiments

We use two datasets for our experiments: MNIST and CelebA.

For MNIST, we take 60,000 images from the *train-images-idx3-ubyte* dataset, splitting them into 48,000 images (80%) for training and 12,000 (20%) for evaluation. All images are rescaled to 32×32 and normalized to $[-1, 1]$ range.

For CelebA, we use the first 100,000 center-aligned images, applying the same 80-20 split as with MNIST. The images are center-cropped to 148×148, resized to 64×64, and then normalized to $[-1, 1]$, following standard practice in the literature.

Experiments are conducted on Runpod Linux machines equipped with RTX3090 GPUs [2]. We implement the models and evaluation metrics using PyTorch 2.6.0 [1] and TorchMetrics 1.6.1 [3]. The model configurations are primarily based on the original papers [7, 5, 17]. Specific model configurations are provided in Table 1. The WGAN-GP architecture follows the DCGAN framework without residual connections [12]. All convolutional (Conv) and fractionally-strided convolutional (FSConv) layers in the hidden layers use 4×4 kernels with a stride of 2 and zero-padding of 1 pixel on each edge. WAE's implementation for CelebA dataset use 5×5 kernels, but we opt for 4×4 for consistency and easier calculation of feature map sizes. Additionally, as recommended in [5], WGAN-GP's critic does not use batch normalization, ensuring that gradients are penalized independently for each input rather than across the entire batch.

In terms of model complexity, VAEs have the fewest trainable parameters, as each layer extracts fewer features. WAE has a comparable number of parameters to VAE for MNIST, but its parameter count increases significantly on CelebA due to a larger latent space and the fully connected discriminator. WGAN-GP has the highest parameter count, with more feature maps in each layer.

Table 1: Model Configurations

Dataset	Model	Latent Dim	Encoder	Decoder/Generator	Discriminator	Num Parameters
MNIST	VAE	32	Conv (BN + ReLU) [32, 32, 64, 64] + Linear ($256 \to 32$)	Linear ($32 \to 64 \times 4 \times 4$) + FSConv (BN + ReLU) [64, 32, 32] + Conv [1]		281377
	WAE	8	Conv (BN + ReLU) [32, 32, 64, 64] + Linear ($256 \to 8$)	Linear ($8 \to 64 \times 4 \times 4$) + FSConv (BN + ReLU) [64, 32, 32] + Conv [1]	Linear (ReLU) $[64] \times 4$ + Linear (Sigmoid) [1]	255530
	WGAN-GP	128		Linear ($128 \to 512 \times 8 \times 8$) + FSConv (BN + ReLU) [512, 256, 128] + Conv [1]	Conv (LeakyReLU) [128, 256, 512] + Linear [1]	9548930
CelebA	VAE	32	Conv (BN + ReLU) [32, 32, 64, 64] + Linear ($256 \to 32$)	Linear ($32 \to 64 \times 4 \times 4$) + FSConv (BN + ReLU) [64, 32, 32] + Conv [3]		433507
	WAE	64	Conv (BN + ReLU) [128, 256, 512, 1024] + Linear ($16384 \to 64$)	Linear ($64 \to 1024 \times 8 \times 8$) + FSConv (BN + ReLU) [512, 256, 128] + Conv [3]	Linear (ReLU) $[512] \times 4$ + Linear (Sigmoid) [1]	28168388
	WGAN-GP	128		Linear ($128 \to 1024 \times 8 \times 8$) + FSConv (BN + ReLU) [1024, 512, 256, 128] + Conv [1]	Conv (LeakyReLU) [128, 256, 512, 1024] + Linear [1]	30635908

4.1 Results

We use a batch size of 64 for all training experiments. The β-VAE models are trained for 100 epochs using the Adam optimizer ($\alpha = 10^{-4}$) with $\beta = 10^{-3}$. WAEs are trained for 100 epochs with a divergence penalty of $\lambda = 1$. The encoder and decoder use a learning rate of 3×10^{-4}, while the learning rate for the discriminator is set to 10^{-3}. WGAN-GP models use a gradient penalty coefficient of $\lambda = 10$. Both the generator and critic are trained with a learning rate of 2×10^{-4}, with the generator updated after every five critic updates. The progression of total loss on training and test sets is shown in Subfigure (a) and (c) in Figure 6, 7 and 8 respectively. VAE loss curves are the smoothest, reflecting stable training. WAE training is stable on MNIST, converging within 20 epochs, but struggles on CelebA. WGAN-GP exhibits smooth training loss progression but volatile test loss behavior. Loss of WGAN-GP is higher in magnitude, but the objective functions are different, and it has higher penalty with gradient mismatch.

During training, we compute the Fréchet Inception Distance (FID) and KL divergence scores (KLD) on the test set every 10 epochs using random samples, with score progressions shown in Subfigures (b) and (d) in Figure 6, 7 and 8 respectively. Since WGAN-GP lacks an encoder, only FID is reported. Table 2 shows a summary of the FID/KL-Divergence scores after the training is done.

For VAEs and WAEs, we evaluate FID using two methods:

- "Reconstruct": The encoder maps an input image x to a latent code z, and the decoder reconstructs $\hat{x}$ based on the latent code z.
- "Random Sample": Latent codes z are randomly sampled from the prior distribution $\mathcal{N}(0, I)$, and the decoder generates synthetic images based on the randomly sampled z.

[2] `www.runpod.io`

As expected, reconstruction-based FID scores are generally lower than random sampling scores since the encoded latent variables retain more information from the input image, enabling better reconstruction.

On MNIST, the WAE encoder maps images to a normal distribution more effectively than β-VAE, as indicated by a lower KL divergence. It also outperforms VAE in reconstruction (lower FID). In random sampling, VAEs and WAEs perform similarly, both surpassing WGAN-GP. Sample generations are shown in Figure 2, while reconstructions are in Figure 3. Notably, WAE-generated digits tend to have thinner strokes and sharper boundaries.

On CelebA, WAE exhibits a significantly higher KL divergence than VAE, likely due to the use of 4×4 kernel rather than 5×5 kernels in the original implementation. The architecture used in our experiments does not match the performance of the original models. Among all models, WGAN-GP achieves the best FID score. Randomly generated CelebA samples are shown in Figure 4. VAE-generated images typically capture only facial structures, with blurry backgrounds and indistinct hair details. Although WAE training is suboptimal, its generated images exhibit clearer boundaries between hair and background with less blurring. WGAN-GP produces the sharpest images with well-defined facial features and backgrounds, though some images appear unnatural. Reconstructions by VAE and WAE are shown in Figure 5, where the blurring effect is more pronounced than random sampling. Some reconstructed images deviate from the original, with anomalies such as added artifacts (*e.g.*, sunglasses appearing in the bottom-left VAE reconstruction).

Table 2: FID/KLD Scores

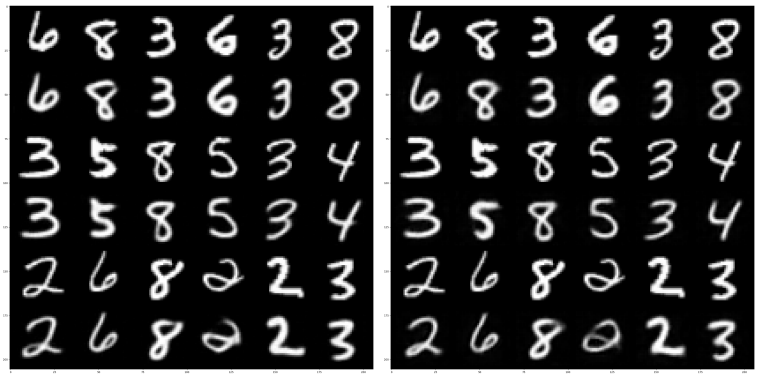
Dataset	Model	KLD	FID (Reconstruct)	FID (Random Sample)
MNIST	VAE	18.00	32.71	31.74
	WAE	8.00	27.45	35.98
	WGAN-GP			68.57
CelebA	VAE	21.95	77.50	83.20
	WAE	61.00	106.60	126.24
	WGAN-GP			37.33

(a) VAE (b) WAE (c) WGAN-GP

Figure 2: MNIST Generation (Random Sample)

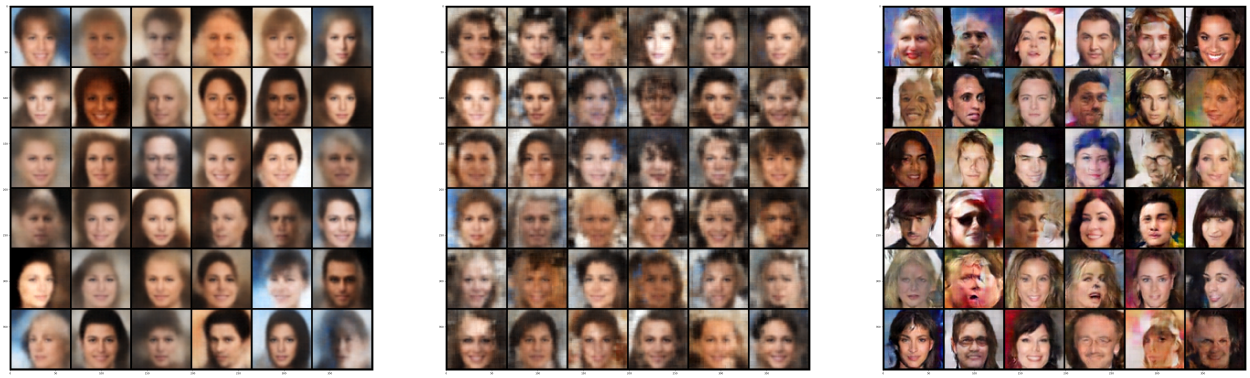
5 Conclusion

This project explores WAEs and compares their mathematical formulation and architecture to VAEs and WGAN-GPs. With its encoder-decoder structure, WAE introduces a conditional independence property between the real image space X and the generated image space Y, allowing the intractable optimal transport cost to be approximated by an additional penalty on the divergence between the prior distribution and the latent representation. When implemented correctly, WAE's encoder provides a closer approximation to the prior distribution and generates higher-quality images. While WAEs typically require more trainable parameters than VAEs due to the additional discriminator, they can achieve better image generation quality (*e.g.*, reduced blurriness) while using fewer parameters than WGAN-GP.



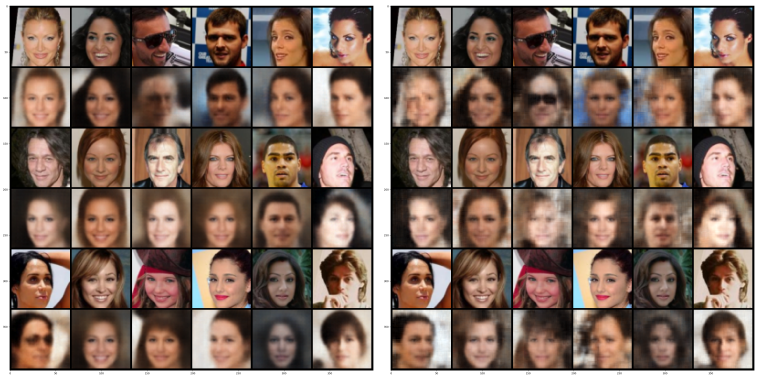
(a) VAE (b) WAE

Figure 3: MNIST Reconstruction. Odd rows are original images, and even rows are reconstructed images.



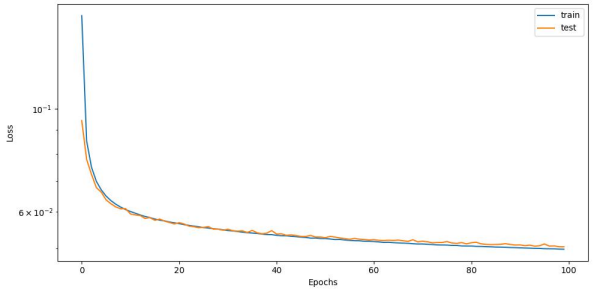
(a) VAE (b) WAE (c) WGAN-GP

Figure 4: CelebA Generation (Random Sample)

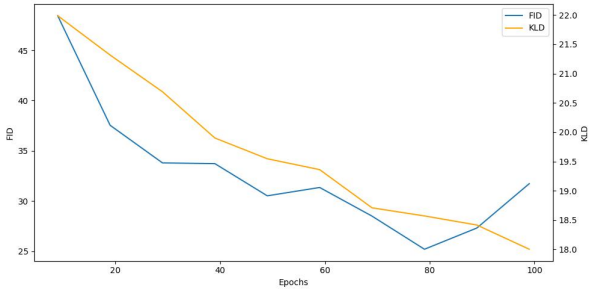


(a) VAE (b) WAE

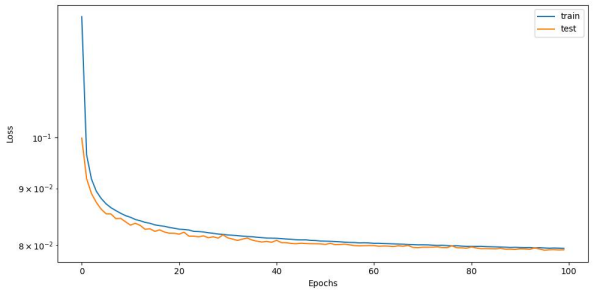
Figure 5: CelebA Reconstruction. Odd rows are original images, and even rows are reconstructed images.



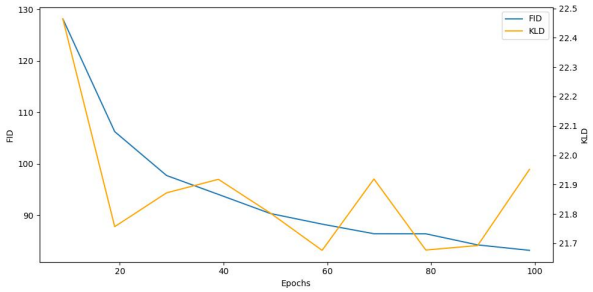
(a) Total Train/Test Loss (MNIST)



(b) FID/KLD on Test Set (MNIST)

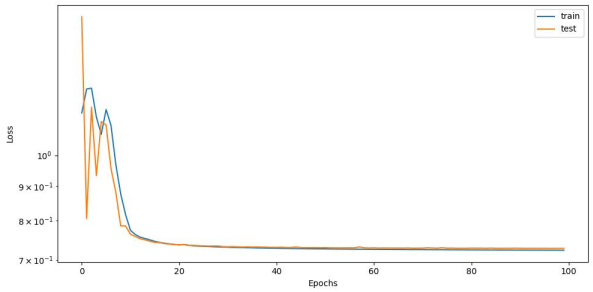


(c) Total Train/Test Loss (CelebA)

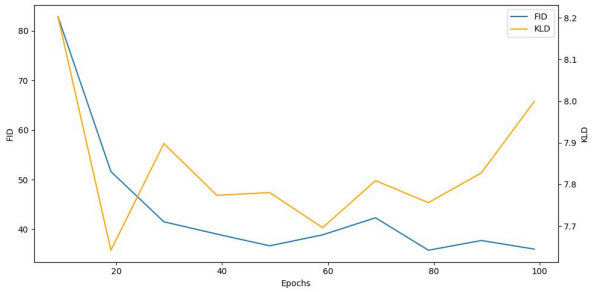


(d) FID/KLD on Test Set (CelebA)

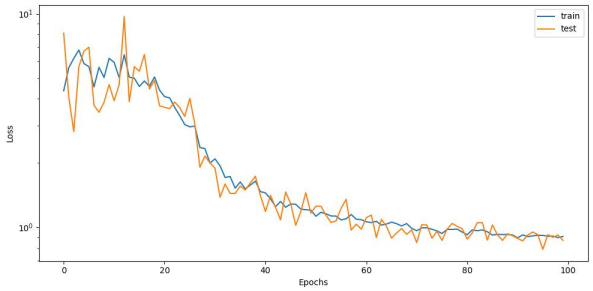
Figure 6: VAE Training Statistics



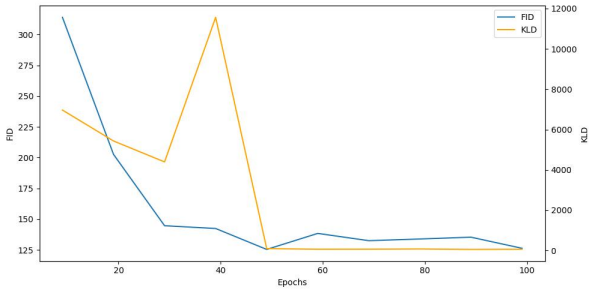
(a) Total Train/Test Loss (MNIST)



(b) FID/KLD on Test Set (MNIST)

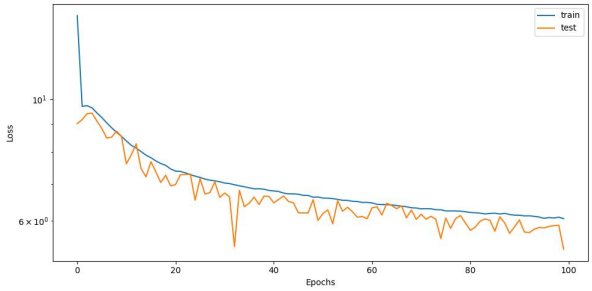


(c) Total Train/Test Loss (CelebA)

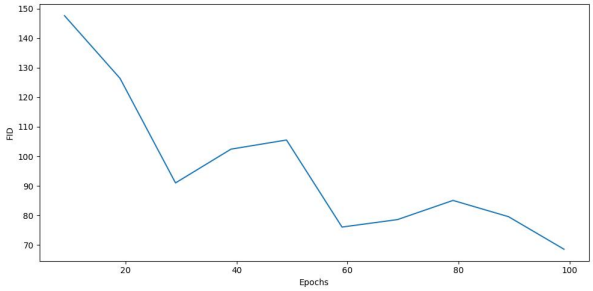


(d) FID/KLD on Test Set (CelebA)

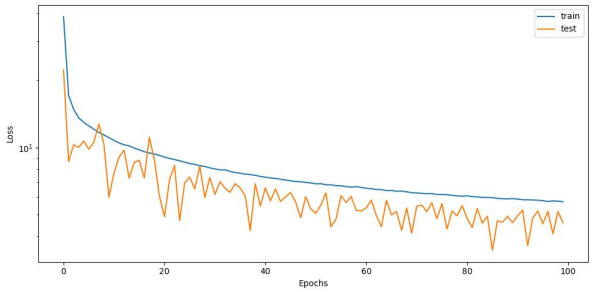
Figure 7: WAE Training Statistics



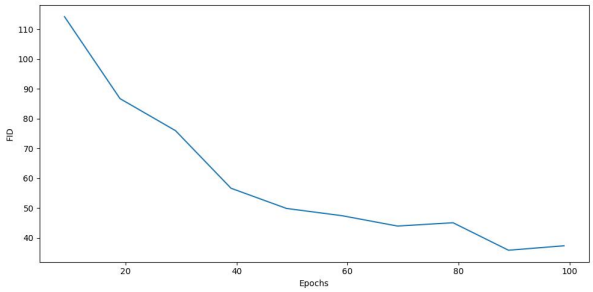
(a) Total Train/Test Loss (MNIST)



(b) FID on Test Set (MNIST)



(c) Total Train/Test Loss (CelebA)



(d) FID on Test Set (CelebA)

Figure 8: WGAN-GP Training Statistics

References

[1] Jason Ansel et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS '24. La Jolla, CA, USA: Association for Computing Machinery, 2024, pp. 929–947. ISBN: 9798400703850. DOI: 10.1145/3620665.3640366. URL: https://doi.org/10.1145/3620665.3640366.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML]. URL: https://arxiv.org/abs/1701.07875.

[3] Nicki Skafte Detlefsen et al. "TorchMetrics - Measuring Reproducibility in PyTorch". In: *Journal of Open Source Software* 7.70 (Feb. 2022). Apache-2.0 License. Code available at https://github.com/Lightning-AI/torchmetrics, p. 4101. DOI: 10.21105/joss.04101. URL: https://www.pytorchlightning.ai.

[4] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[5] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG]. URL: https://arxiv.org/abs/1704.00028.

[6] Martin Heusel et al. "GANs trained by a two time-scale update rule converge to a local nash equilibrium". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6629–6640. ISBN: 9781510860964.

[7] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[9] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: (2013). arXiv: 1312.6114 [stat.ML]. URL: http://arxiv.org/abs/1312.6114.

[10] Alireza Makhzani et al. *Adversarial Autoencoders*. 2016. arXiv: 1511.05644 [cs.LG]. URL: https://arxiv.org/abs/1511.05644.

[11] Tristan Milne and Adrian I Nachman. "Wasserstein GANs with Gradient Penalty Compute Congested Transport". In: *Proceedings of Thirty Fifth Conference on Learning Theory*. Ed. by Po-Ling Loh and Maxim Raginsky. Vol. 178. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 103–129. URL: https://proceedings.mlr.press/v178/milne22a.html.

[12] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG]. URL: https://arxiv.org/abs/1511.06434.

[13] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Vol. 87. Progress in Nonlinear Differential Equations and Their Applications. Springer, 2015. DOI: 10.1007/978-3-319-20828-2.

[14] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG]. URL: https://arxiv.org/abs/2011.13456.

[15] Jan Stanczuk et al. *Wasserstein GANs Work Because They Fail (to Approximate the Wasserstein Distance)*. 2021. arXiv: 2103.01678 [stat.ML]. URL: https://arxiv.org/abs/2103.01678.

[16] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV]. URL: https://arxiv.org/abs/1512.00567.

[17] Ilya Tolstikhin et al. *Wasserstein Auto-Encoders*. 2019. arXiv: 1711.01558 [stat.ML]. URL: https://arxiv.org/abs/1711.01558.