Model Collapse in Wasserstein GANs and VAEs

Carl Fredrik Nordbø Knutsen,[1] Ben René Bjørsvik,[1] and Jonas Båtnes[1]

[1] *University of Oslo, Department of Physics*

(Dated: June 7, 2025)

Generated data makes for an ever-increasing proportion of data found on the internet. This fact raises the important question: how do machine learning models perform when trained on data generated by other machine learning models? We explore this question in the particular setting of image generation, on the MNIST dataset. We train 20 generations of Wasserstein Generative Adversarial Networks, each trained on data generated by the previous generation. We also perform this recursive training procedure for 20 generations of Variational Autoencoders. The Variational Autoencoders experience model collapse, and degrade to only producing a single type of image. The Wasserstein Generative Adversarial Networks appear slightly more robust to collapse in mode, however the generated image quality clearly falls for later generations. Overall, we find clear indications of model collapse, highlighting the danger of training on artificial data.

I. Introduction

Artificial intelligence (AI) is a rapidly evolving discipline, distinguished by its capacity to tackle complex problems that exceed the reach of traditional computational methods. Within this landscape, generative modeling has emerged as a particularly transformative area, enabling machines to synthesize realistic data such as images, audio, and text with applications spanning from creative industries to scientific research [1]. Central to this progress are frameworks like Generative Adversarial Networks (GANs), Wasserstein GANs (WGANs), and Variational Autoencoders (VAEs), each offering distinct strategies for approximating data distributions.

GANs, introduced by Goodfellow et al. in 2014 [1], initiate a two-player game between a generator and a discriminator, whereby the generator learns to produce plausible data samples, and the discriminator attempts to distinguish these from real examples. This adversarial setup has achieved impressive results in synthetic data generation but is also prone to training pathologies such as mode collapse, where the generator converges to producing only a narrow subset of the target distribution [2]. To address such instabilities, Arjovsky et al. proposed WGANs [3], which replace the standard loss with a Wasserstein distance metric grounded in optimal transport theory. This approach yields improved convergence behavior and training robustness, and was further refined by Gulrajani et al. through the introduction of gradient penalties (WGAN-GP) [4]. In contrast, VAEs, introduced by Kingma and Welling [5], employ a probabilistic generative process that encodes data into a continuous latent space and reconstructs it through variational inference, allowing for structured latent representations and efficient sampling.

Mode collapse, is a phenomenon which arises during training of a single model, where the model fits to a single mode in the target distribution. A pressing issue in the development of generative models is the phenomenon of *model collapse*. In contrast with mode collapse, model collapse refers to a progressive degradation that occurs when training new generations of models on data generated by previous generations [6]. This recursive self-training leads to a gradual loss of diversity and fidelity in generated data, ultimately distorting the learned distribution and decoupling it from the original dataset, a dynamic also referred to as "AI autophagy" [7]. Understanding and mitigating this phenomenon is crucial as generative models are increasingly trained on data sourced from the internet, which to an increasing extent consists of AI-generated content.

In this paper, we undertake a detailed examination of WGANs with gradient penalty and VAEs. Our study maintains a focus on the long-term effects of recursive training and the onset of model collapse. We begin by outlining the theoretical foundations of these generative architectures and describe our implementation methodology. Next, we train initial WGAN and VAE models on the MNIST dataset [8]. We proceed to recursively train models based on previous generated data. To assess model degradation across generations, we conduct both qualitative and quantitative evaluations: visual inspection of image quality, analysis of pixel intensity distributions, and classification of generated images using an independently trained Convolutional Neural Network (CNN). We further quantify divergence from the original data distribution using Wasserstein-1 distance and Fréchet Inception Distance. Finally, we assess the resilience of WGAN discriminators by measuring their classification accuracy on real and generated data.

Through this study, we aim to clarify how generative models evolve over recursive training and highlight model collapse as a central challenge for the sustainable development of generative AI systems. All source code and experiment results presented in this work are available on our GitHub repository: https://github.com/carlfre/wasserstein.git.

II. Methods

The generative models GANs, WGANs, and VAEs represent distinct but powerful frameworks for learning complex data distributions. Their theoretical foundations and practical implementations leverage different mathematical principles, leading to varying strengths and weaknesses in terms of training stability, sample quality, and interpretability. In the following, we outline the methodology for each, progressing from their core architectural components and objective functions to specific optimization techniques and common challenges.

A. Generative Adversarial Networks (GANs)

GANs, introduced by Goodfellow et al. [1], frame the generative modeling problem as a two-player minimax game between two neural networks: a generator G and a discriminator D. The generator's objective is to learn the data distribution p_{data} and generate synthetic samples that are indistinguishable from the real data. The discriminator's role is to estimate the probability that a given sample comes from the real data distribution rather than the generator's distribution p_g.

Formally, the generator $G(z;\theta_g)$, parameterized by θ_g, takes a sample z from a predefined prior noise distribution $p_z(z)$ (commonly a uniform distribution over a hypercube or a standard Gaussian distribution) and maps it to a point in the data space, thereby implicitly defining the generative distribution p_g. The discriminator $D(x;\theta_d)$, parameterized by θ_d, receives an input x (either a real data sample or a generated sample) and outputs a single scalar probability $D(x) \in [0,1]$, representing the likelihood that x originates from p_{data}.

The training process is an adversarial game formulated as a minimax objective function $V(D,G)$:

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

The discriminator aims to maximize $V(D,G)$, which corresponds to correctly classifying real samples and generated samples. To correctly classify a real sample, $D(x)$ should be close to 1 for a real sample x. Whereas $D(G(z))$ should be close to 0 for a generated image $G(z)$. Simultaneously, the generator aims to minimize $V(D,G)$, which is equivalent to maximizing $\log D(G(z))$, thereby fooling the discriminator into classifying generated samples as real. This adversarial process drives the generator to produce samples that increasingly resemble the true data distribution. This process is illustrated in fig. 1.

At a fixed generator G, the optimal discriminator $D^*_G(x)$ is given by:

$$D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \tag{1}$$

This indicates that the optimal discriminator acts as a probability ratio, comparing the likelihood of x under the real data distribution versus the generated distribution. The value of the objective function at this optimum, $V(D^*_G, G)$, can be shown to be directly related to the Jensen-Shannon (JS) divergence [9] between p_{data} and p_g:

$$\max_D V(D,G) = -\log 4 + 2 \cdot JSD(p_{\text{data}} || p_g)$$

Minimizing this value with respect to G thus corresponds to minimizing the JS divergence between the data distribution and the generator's distribution.

Ideally, the model converges such that $p_g = p_{\text{data}}$. At this equilibrium, the optimal discriminator $D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_{\text{data}}(x)} = \frac{1}{2}$ everywhere, meaning the discriminator cannot distinguish between real and fake samples.

However, training standard GANs is chronically challenging. Common issues include [9]:

- **Mode Collapse:** The generator produces a limited variety of samples, collapsing to a few modes of the data distribution, rather than capturing the full diversity. This is often attributed to the generator finding a single type of output that consistently fools the current discriminator, which then fails to provide gradients that encourage exploration of other modes.
- **Vanishing/Exploding Gradients:** Similar to other deep learning models, issues with gradients can arise, particularly in deep architectures.
- **Training Instability:** The adversarial game can be difficult to balance. If the discriminator becomes too strong relative to the generator, gradients can vanish, providing little information for the generator to learn.

The reliance on the JS divergence, which is discontinuous for disjoint supports (a common scenario early in GAN training, especially for high-dimensional data), contributes further to training instability and vanishing gradients.

B. Wasserstein GANs (WGANs)

WGANs [3] were introduced to address the training instability and mode collapse issues of traditional GANs by replacing the JS divergence with the Earth-Mover (EM) distance, also known as the Wasserstein-1 distance. The EM distance is a metric that quantifies the minimum "cost" of transforming one probability distribution into another, where the cost is typically the total amount of "mass" that needs to be moved multiplied by the distance it is moved. This can be intuitively thought of as the minimum work required to move a pile of earth shaped

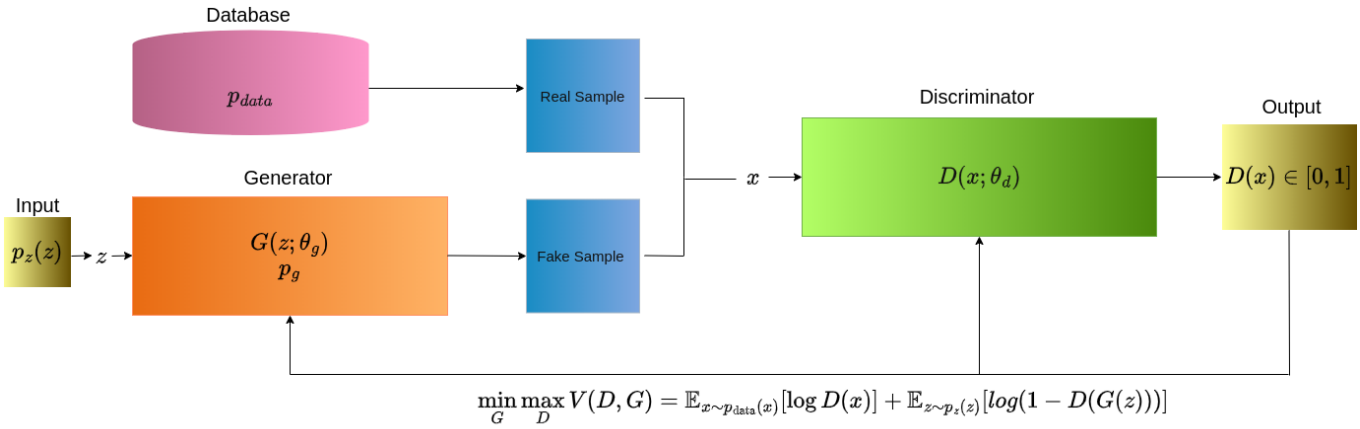


FIG. 1: Illustration of the basic architecture of a GAN. Below we see the optimization problem that we are trying to solve.

like one distribution to form a pile shaped like another distribution, hence the name "Earth-Mover" distance.

Formally, the EM distance between two probability distributions $\mathbb{P}_r$ (real data distribution) and $\mathbb{P}_g$ (generator distribution) is defined as:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[|x - y|] \tag{2}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$. Each $\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)$ can be interpreted as a transport plan, specifying how much mass to move from point x to point y. The infimum is taken over all possible transport plans to find the minimum cost.

A key advantage of the EM distance is that it is continuous and differentiable almost everywhere, even when the supports of the two distributions are disjoint. This property provides much more stable gradients compared to the JS divergence used in standard GANs, particularly when the generator's output distribution is far from the real data distribution.

Directly computing the infimum in the definition of the EM distance is generally hard. However, the Kantorovich-Rubinstein duality [10] provides a possible way to compute the EM distance:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{f:|f|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)] \tag{3}$$

where the supremum is taken over all 1-Lipschitz functions f. A function f is 1-Lipschitz if for all x_1, x_2, $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$. This condition limits the rate of change of the function.

In WGANs, the discriminator is replaced by a "critic" f, which is trained to approximate the 1-Lipschitz function that maximizes the right-hand side of eq. (3). The critic's objective is to maximize the difference between the average score of real samples and the average score of generated samples. The generator's objective is then to minimize this estimated EM distance by generating samples that receive high scores from the critic. The objective function for WGAN becomes:

$$\min_G \max_{f:|f|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{z\sim p_z(z)}[f(G(z))]$$

The 1-Lipschitz constraint on the critic f is crucial for ensuring that the quantity being optimized is indeed the EM distance.

The original WGAN paper proposed enforcing the 1-Lipschitz constraint by clipping the weights of the critic network to a small range, typically $[-c, c]$ for some small $c \in \mathbb{R}$. However, this weight clipping method has several limitations [3]:

- **Capacity Underuse:** Clipping weights can restrict the capacity of the critic network, preventing it from fully learning the optimal 1-Lipschitz function.
- **Gradient Issues:** If the clipping range c is too large, it may take a long time for weights to reach the boundaries, effectively rendering the constraint weak. If c is too small, it can lead to vanishing or exploding gradients, especially in deeper networks, and concentrate the learned function to a narrow range around 0, limiting its expressiveness.
- **Sensitivity to c:** The performance of WGAN with weight clipping is highly sensitive to the choice of the clipping parameter c.

To overcome these limitations, WGANs with Gradient Penalty (WGAN-GP) [4] were introduced. WGAN-GP enforces the 1-Lipschitz constraint by adding a penalty to the critic's loss function that penalizes the deviation of the gradient norm of the critic's output with respect to its input from 1. The intuition is that for a function to be 1-Lipschitz, its gradient norm should be at most 1

everywhere. Penalizing the deviation from 1 encourages the gradient norm to be close to 1, particularly in the region between the real and generated data distributions, where the critic's function needs to distinguish between the two.

The gradient penalty is applied to interpolated samples $\hat{x}$ drawn uniformly along straight lines between pairs of points (x_r, x_g), where $x_r \sim \mathbb{P}_r$ and $x_g \sim \mathbb{P}_g$. The loss function for the critic in WGAN-GP is

$$L(f) = \mathbb{E}_{x_g \sim \mathbb{P}_g}[f(x_g)] - \mathbb{E}_{x_r \sim \mathbb{P}_r}[f(x_r)] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} f(\hat{x})||_2 - 1)^2]. \quad (4)$$

The first two terms represent the estimated Earth-Mover distance. The third term is the gradient penalty, where $\nabla_{\hat{x}} f(\hat{x})$ is the gradient of the critic's output with respect to the interpolated sample $\hat{x}$, $||\cdot||_2$ is the L2 norm, and λ is a penalty coefficient (typically set to 10). The generator is then trained to minimize $-\mathbb{E}_{z \sim p_z(z)}[f(G(z))]$. WGAN-GP significantly improves training stability and generates higher-quality samples compared to the original WGAN with weight clipping, making it a more robust approach.

C. Variational Autoencoders (VAEs)

VAEs [5, 11] provide a probabilistic framework for generative modeling by explicitly modeling the data distribution $p_\theta(x)$ through a latent variable model. Unlike GANs, which learn an implicit mapping from noise to data, VAEs define a directed graphical model for data x with latent variables z. The generative process assumes that data is generated by first sampling a latent variable z from a prior distribution $p(z)$ and then sampling the data x from a conditional distribution $p_\theta(x|z)$. The prior $p(z)$ is typically chosen to be a simple distribution, such as a standard multivariate Gaussian $\mathcal{N}(0, I)$. The conditional distribution $p_\theta(x|z)$ is modeled by a decoder network, parameterized by θ, which takes z as input and outputs parameters of the distribution over x (e.g., mean and variance for a Gaussian likelihood, probabilities for a Bernoulli likelihood).

The primary goal in VAEs is to maximize the marginal likelihood of the data, $p_\theta(x) = \int p_\theta(x|z)p(z)dz$. However, this integral is often hard to solve for complex data distributions and decoder networks. VAEs address this by introducing an approximate posterior distribution $q_\phi(z|x)$, modeled by an encoder network parameterized by ϕ. The encoder takes data x as input, and outputs estimated parameters of the distribution over the latent variable z (e.g., mean and variance for an approximate posterior Gaussian). An illustration of the VAE architecture is given in fig. 2.

Training VAEs involves maximizing a lower bound on the marginal likelihood, known as the Evidence Lower Bound (ELBO) or negative Free Energy. The ELBO for a single data point x is derived using Jensen's inequality:

$$\begin{aligned}
&\log p_\theta(x) = \log \int p_\theta(x|z)p(z)\, dz \\
&= \log \mathbb{E}_{p(z)}[p_\theta(x|z)] \\
&= \log \mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] \\
&\geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] \text{ (by Jensen's inequality)} \\
&= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p(z)}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))
\end{aligned}$$

The ELBO,

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})),$$

consists of two terms:

- **Reconstruction Term:** $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$. This term measures how well the decoder can reconstruct the input data x from a latent sample z drawn from the approximate posterior $q_\phi(z|x)$. Maximizing this term encourages the decoder to learn a mapping from the latent space back to the data space that can accurately reproduce the observed data. It acts as a measure of reconstruction accuracy under the learned latent representation.
- **KL Divergence Term:** $-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$. This term penalizes the difference between the approximate posterior $q_\phi(z|x)$ and the prior distribution $p(z)$. Minimizing this term encourages the encoder to map the input data x to a latent distribution $q_\phi(z|x)$ that is close to the prior $p(z)$. This acts as a regularizer, ensuring that the latent space is structured in a way that is easy to sample from (following the prior) for generation.

The total objective for a dataset $D = \{x^{(i)}\}_{i=1}^N$ is to maximize the sum of the ELBOs for each data point:

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \quad (5)$$

This objective is typically optimized using stochastic gradient descent on minibatches of data.

A crucial aspect of training VAEs is backpropagating gradients through the sampling process from the approximate posterior $q_\phi(z|x)$. If $q_\phi(z|x)$ is a deterministic function of ϕ, this is straightforward. However, when $q_\phi(z|x)$ is a distribution with parameters output by the encoder, the sampling operation is not differentiable. The reparameterization trick [5, 11] addresses this by expressing the random sample z as a deterministic function of its parameters and an auxiliary random variable with a fixed distribution. For a Gaussian approximate posterior

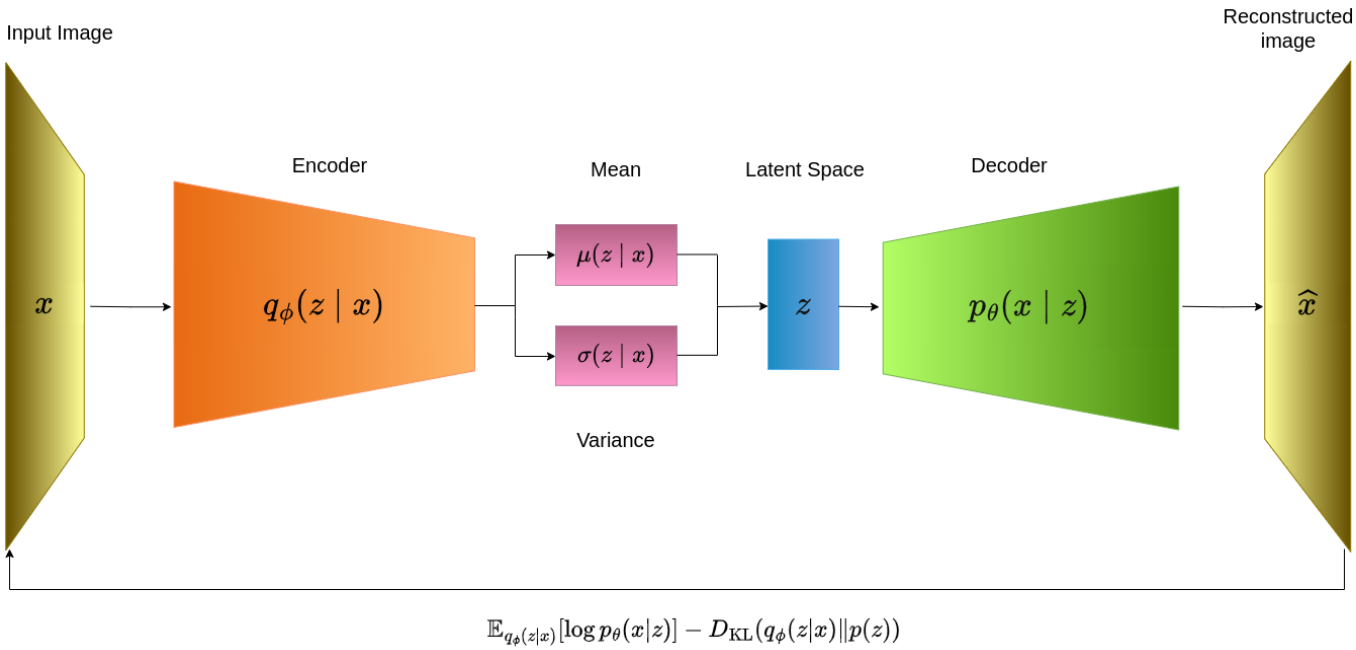


FIG. 2: Illustration of the basic architecture of a VAE with objective function at the bottom.

$q_\phi(z|x) = \mathcal{N}(z; \mu(x;\phi), \sigma^2(x;\phi))$, where μ and σ^2 are the mean and variance vectors output by the encoder, a latent sample z can be obtained by:

$$z = \mu(x;\phi) + \sigma(x;\phi) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{6}$$

where $\odot$ denotes element-wise multiplication, and ϵ is a sample from a standard Gaussian distribution. This allows the gradients with respect to ϕ to be backpropagated through μ and σ^2.

For the common case of a Gaussian approximate posterior and a standard Gaussian prior, the KL divergence term has a closed-form solution:

$$D_{KL}(q_\phi(z|x)||p(z)) = \frac{1}{2}\sum_{j=1}^{J}\left(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1\right) \tag{7}$$

where J is the dimensionality of the latent space, μ_j and σ_j^2 are the j-th components of the mean and variance vectors μ and σ^2.

While VAEs offer stable training compared to GANs and are less prone to mode collapse due to optimizing a lower bound on the data likelihood, they can suffer from "posterior collapse." This occurs when the KL divergence term dominates the ELBO, forcing the approximate posterior $q_\phi(z|x)$ to match the prior $p(z)$ regardless of the input data x. In such cases, the latent code z becomes independent of x, and the decoder learns to generate data from the prior $p(z)$ alone, ignoring the input and potentially generating low-quality or uninformative samples. Strategies to mitigate posterior collapse include annealing the KL term during training, modifying the network architecture, or using more expressive approximate posterior distributions.

The choice of the decoder's output distribution $p_\theta(x|z)$ depends on the type of data being modeled. For continuous data like images, a multivariate Gaussian or a more flexible distribution like a Laplacian or a mixture of Gaussians is often used. For discrete data, such as text or binary images, a categorical or Bernoulli distribution is appropriate. The VAE framework provides a principled way to perform inference and generation by explicitly modeling the latent variable space. While generated samples might sometimes be blurrier than those from state-of-the-art GANs, VAEs offer better interpretability of the latent space and a clear probabilistic formulation.

D. Fréchet inception distance (FID)

In practice, evaluating the quality of generated images remains a critical challenge. While traditional GAN metrics like discriminator loss or visual inspection provide some insight, they are often insufficient for reliable model comparison. A widely adopted quantitative metric is the Fréchet Inception Distance (FID) [12], which measures the similarity between real and generated data distributions using the statistics of feature representations extracted from a pretrained Inception network. Specifically, real and generated images are passed through the Inception-v3 model, and the resulting activations are modeled as multivariate Gaussians with means μ_r, μ_g and covariances Σ_r, Σ_g for real and generated data, re-

spectively. The FID is then computed as:

$$\text{FID} = ||\mu_r - \mu_g||_2^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \quad (8)$$

In fact, the FID coincides with the Wasserstein-2 distance, also known as the Fréchet distance, between the estimated normal distributions [12]. Lower FID values indicate closer alignment between the two distributions, implying similar fidelity and diversity of the generated images compared to that of the real images. Unlike other metrics, FID is sensitive to both the quality and variation of generated samples and is robust to small image perturbations, making it a preferred choice for benchmarking modern generative models such as WGANs and VAEs.

E. Experiment Setup

1. The algorithm

The algorithm we used for recursive training was inspired by the work of Shumailov et al. [6]. It assumes that the WGAN and VAE architectures are already created such that they can be easily instantiated in the algorithm. In this project we call this the Bigloop algorithm, which is presented in algorithm 1.

Algorithm 1 The Bigloop Iterative Training Algorithm

Require: N_{gen}, Number of generations to train.
S_{data}, Size of the dataset to generate at each step.
D_{real}, The initial real-world dataset.
Initialize $D_0 \leftarrow D_{real}$
for $i = 1$ **to** N_{gen} **do**
  // Initialize new model for the current generation
  $M_i \leftarrow$ InitializeModel(type $\in$ {VAE, WGAN})
  // Train new model on data from previous generation
  $M_i, \text{loss}_i \leftarrow \text{TrainModel}(M_i, D_{i-1})$
  // Save trained model and its final loss
  SaveModel(M_i)
  RecordLoss(loss_i)
  // Generate new dataset using newly trained model
  $D_i \leftarrow \text{GenerateData}(M_i, S_{data})$
end for
return All saved models and recorded losses.

2. Model setups

In our WGAN implementation we created a generator consisting of 4 transposed 2D convolutional layers with LeakyReLU as the activation function for the 3 first layers and Tanh as the activation function for the last layer. In order to stabilize the training and to increase the quality of the generated images, we implemented instance normalization between all the layers with scaling factors 256, 128 and 64 respectively. The discriminator are created exactly opposite of the generator, except that the last layer used a Sigmoid activation function which outputs a number between 0 and 1 to classify whether the sample is real or fake. This is illustrated in fig. 3, together with the data dimensions in each layer.

For the WGAN-hyperparameters we chose to test two different setups. Both setups had a batch size of 64, a learning rate of 0.0002, and a gradient penalty coefficient (Lambda GP) of 10. Setup 1 trained for 100 epochs with 5 training iterations, whereas setup 2 trained for 300 epochs with 2 critic iterations. The choice of having 100 and 300 epochs was to give the discriminator approximately the same volume of training for each number of critic iterations.

In our VAE implementation we created an encoder consisting of 2 hidden layers with LeakyReLU as activation functions. The output layer are the mean and variance calculations which result in the latent variable z. The decoder was implemented exactly the opposite of the encoder, except that a sigmoid activation function is applied as the last layer before the output is reshaped to have dimensions 28×28. This is illustrated in fig. 4, together with the data dimensions in each step.

We trained our WGANs and VAEs over 20 generations. For each generation, we initialized a new WGAN or VAE and trained it on data generated by the previous generation, except the first iteration in which the models were trained on the original MNIST data set.

The specific training parameters are presented in table I.

TABLE I: Training parameters for WGAN and VAE.

	WGAN		*VAE*
Parameter	**Setup 1**	**Setup 2**	**-**
Batch Size	64	64	64
Epochs	100	300	100
Learning Rate	0.0002	0.0002	0.001
Lambda GP	10	10	N/A
Critic Iterations	5	2	N/A

III. Results and Discussion

We trained 20 generations of VAE models and WGAN models using algorithm 1, with the training parameters described in table I. We emphasize that the WGAN generator is trained significantly more for Setup 2, as compared to Setup 1. First we will analyze the quality of the generated images. We will for instance consider some measures of distributional distances, comparing the generated data with the original MNIST data. Afterwards we will assess the performance of the WGAN discriminators on real and generated data, and discuss how these performance relates to potential model collapse.

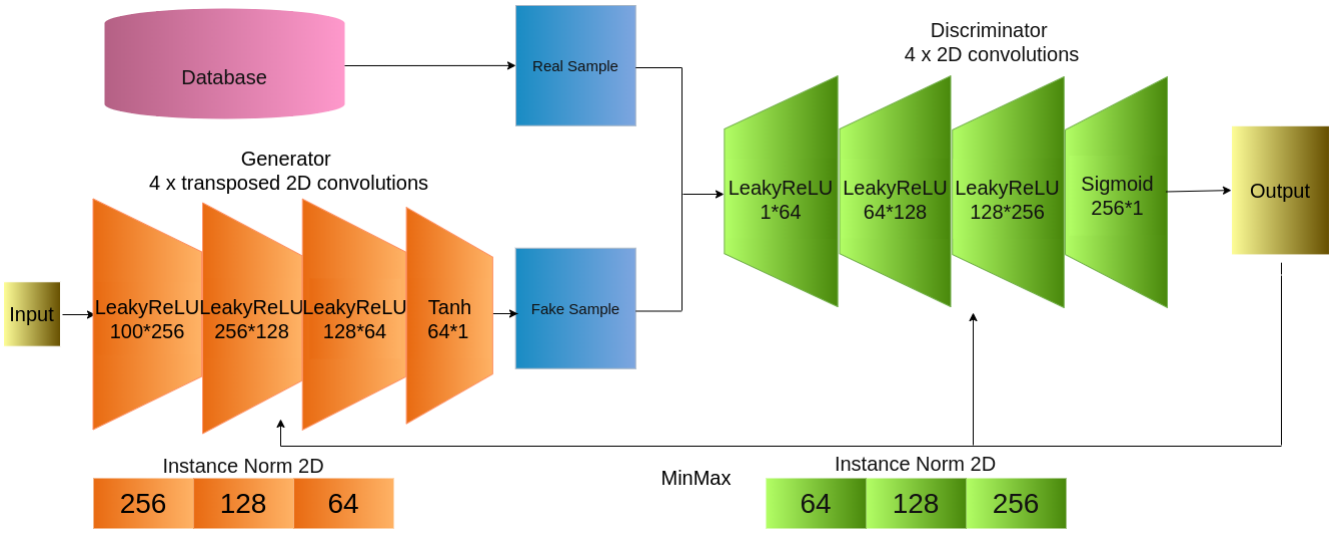


FIG. 3: Illustration of our specific implementation of WGAN.

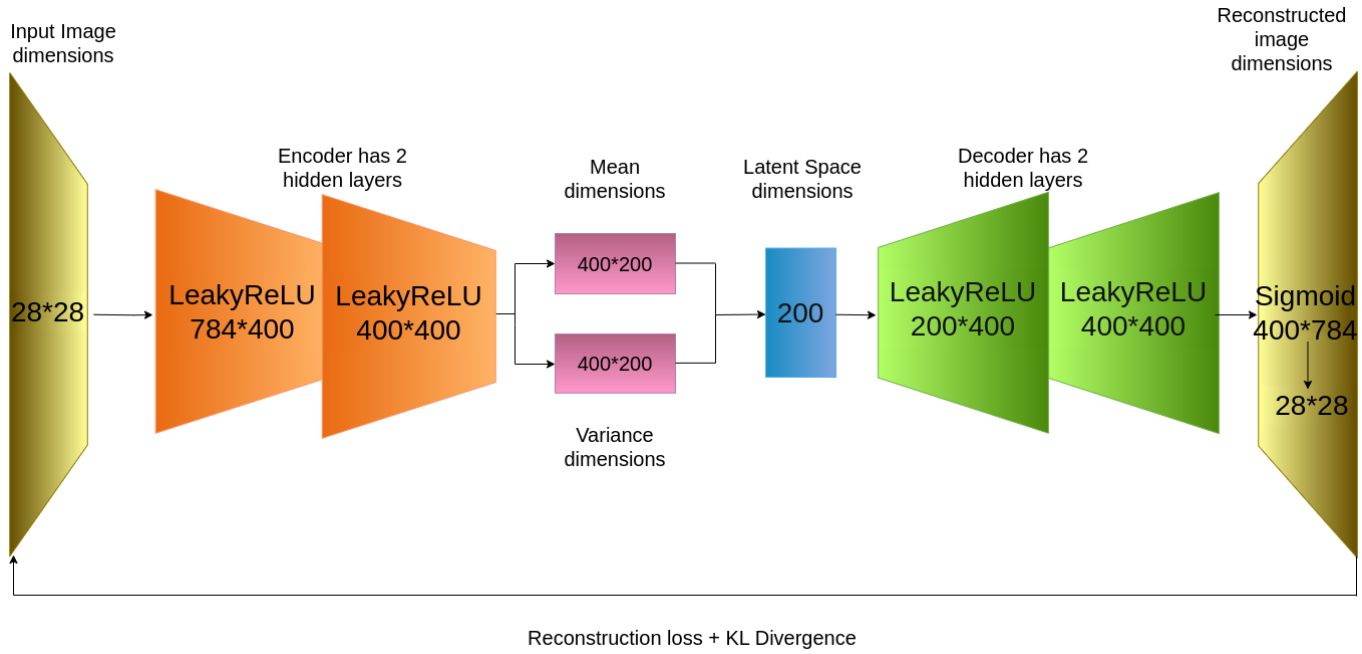


FIG. 4: Illustration of our specific implementation of VAE with objective function at the bottom.

A. Quality of Generated Images

We generate example images using the WGAN generator (for Setups 1 and 2) and the VAE decoder, for generations 1, 5, 10, and 20. The images can be seen in fig. 5.

There is a qualitative reduction in image quality with increasing generations, for all models. The images generated by WGAN Setup 1 become noisier, and resemble real numbers less and less. We see a similar trend for Setup 2, however the images have slightly higher overall quality. The noisiness and jaggedness present in the WGAN-generated images, are not at all present in the VAE-generated images. Instead, the later generations of the VAE decoder appear to generate images with washed out details.

There is a clear indication of model collapse in the VAE-generated images. There is a marked reduction in image variety by generation 10. By generation 20, similar images are generated for the whole 5x5 grid. The numbers all look like something between a 5, 8, or 9. This finding is consistent with those in the supplementary material of [6]. The produced images could represent something like an average of the images in the original dataset. For the WGAN models, the indication of model collapse is not

as strong. for Setup 1, there does appear to be quite a few images that look like 7's, and for Setup 2 there are quite a few 1's. This by itself is, however, not a strong enough effect to conclude that the WGAN models experience model collapse.

Next, we seek to find out which numbers our models generate images of (to the extent that the generated images actually resemble numbers). We proceed to train a CNN [13] on classifying images in the MNIST dataset, achieving a 99.25% accuracy score. We make the assumption that the CNN classifier will also be able to accurately classify the generated images, and apply it to datasets generated by the VAE and WGAN models. Relative frequencies of predicted classes are shown in fig. 6. Both for the VAE and WGAN Setup 2, certain predicted classes become way more dominant for later generations. This signals that we get less varied generated data for later generations of the VAE and WGAN Setup 2, which is an indication of model collapse. Of course this is entirely in line with what we see in the images in fig. 5. We do not see a similarly clear pattern for WGAN Setup 1, though the classes 2 and 3 appear to become more prevalent in the latest generations.

However, in order for the CNN classification to give us any information about the images, we have made the strong assumption that the CNN generalizes to classifying images outside of the MNIST dataset. In fact, we see some indications that this is not the case - the CNN appears to perform poorly on the generated data. The clearest sign is that the most prevalent classification made on the VAE-generated data is the number 1. We qualitatively judge that the images generated in the later generations, as seen in fig. 5, do not particularly resemble the number 1. We have also observed similarly poor classifications on the WGAN-generated data. Nevertheless, we interpret certain classes becoming more dominant as a sign that the generated datasets might become more homogeneous in the later generations.

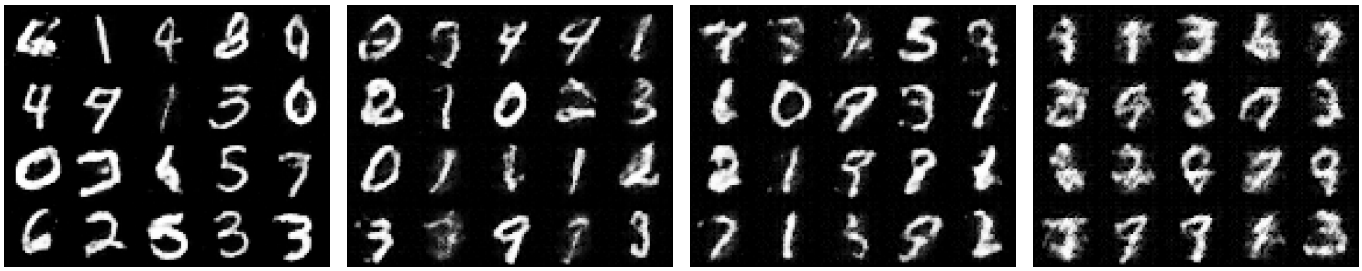
Next, we consider the distribution of pixel intensities, as a simple heuristic for distinguishing datasets. The pixel intensity distributions for images generated by generations 1, 10, and 20 of the different models, as well as the intensity distribution in the MNIST dataset, can be seen in fig. 7. The pixel distributions of the generated datasets are clearly distinct from the intensity distribution of the MNIST images, already from the first generation. The most noticable difference is the smoothness of the generated pixel distributions, as compared with the discrete spikes apparent in the MNIST pixel distribution. This discrepancy is likely due to the differentiability of neural networks, making it hard to output discrete values without having a similar probability density for the neighbouring pixel intensities. Furthermore, we remark that the WGAN models output negative pixel intensities - in this instance, this is a clear way of distinguishing generated images from real ones.

As a general trend, the earlier generations have intensity distributions that resemble the ground truth pixel distribution the most. The first generation models have the sharpest probability density peaks around 0, similar to the distribution in the MNIST images. For the later generations shown in fig. 7, the pixel intensity distributions tend to spread out. Training on generated data appears to have an effect on the pixel intensity distribution that is similar to applying a gaussian filter - this resemblance is most clear for the WGAN models. The fact that later generations have lower and more rounded peaks, shows one way in which our models fail to accurately learn the original image distribution. We also note that, particularly for generation 20 of the VAE, a lot of probability mass has been moved to the middle of the pixel intensity range.

We will now consider two quantitative measures of distances between distributions, namely the Wasserstein-1 distance and the FID. The Wasserstein-1 distance and the FID between generated images and MNIST images are both shown in fig. 8, as a function of generation number.

The Wasserstein-1 distance between a generated dataset of size 500 and a random selection of 500 distinct images from the MNIST dataset is plotted as a function of model generation in fig. 8(a). The metric is calculated after flattening all images, treating them as 1-dimensional vectors of size 784 (28·28). Due to the computational complexity of the distance calculations we restricted ourselves to using only 500 images from each dataset. From the figure we can tell that the distance evolves in somewhat different fashions depending on the model, but the general trend is an increasing distance over generations. We might question what effect the flattening of the images has on the metrics. When flattening the images we surely lose some important spatial structure in the images. However, all the images, as illustrated in fig. 5, have the same typical structure with centered digits with a black surrounding background. Therefore the flattened images may still be comparable, as we can expect the pixels of the background and those of the digits to be distributed similarly along the vectorized images. Given that the flattened images still contain important structural information about the images, the trend of increasing distances may be, to a certain degree, representative of a similar trend in the general dataset, namely that the generated datasets over generations diverge from the reference MNIST-dataset.

The FID distance between generated data and the ground truth is shown in fig. 8(b). A similar trend, to the one apparent in the Wasserstein distance plots, is shown. The FID estimation was performed using a pretrained neural net provided in the PyTorch framework. The network requires the images to be of size $3 \times 299 \times 299$, i.e. 3 channels and length and width of 299 pixels, which was solved in a preprocessing step of extending and upscaling the $1 \times 28 \times 28$-images from the generated and MNIST

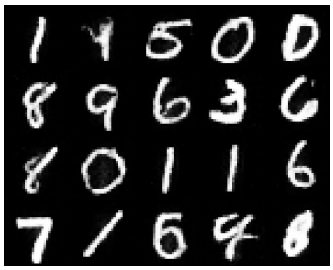


(a) Generation 1 WGAN. (Setup 1)

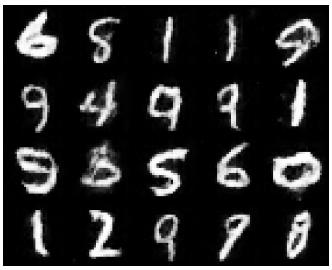
(b) Generation 5 WGAN. (Setup 1)

(c) Generation 10 WGAN. (Setup 1)

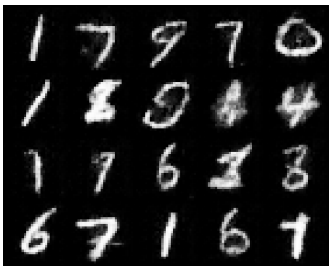
(d) Generation 20 WGAN. (Setup 1)



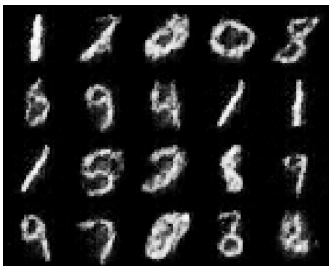
(e) Generation 1 WGAN. (Setup 2)



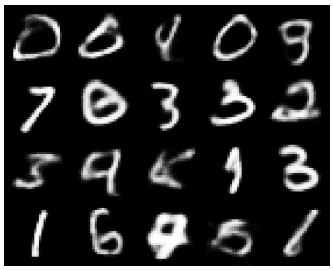
(f) Generation 5 WGAN. (Setup 2)



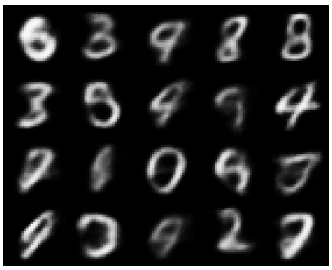
(g) Generation 10 WGAN. (Setup 2)



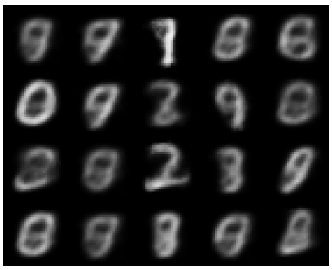
(h) Generation 20 WGAN. (Setup 2)



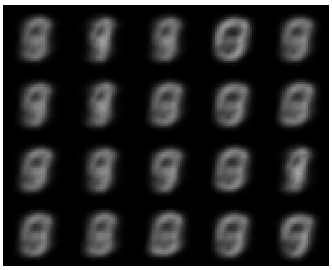
(i) Generation 1 VAE.



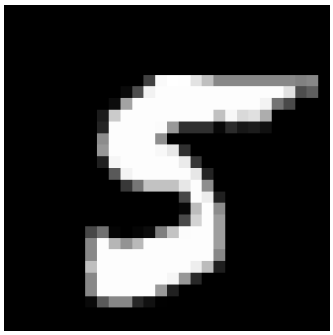
(j) Generation 5 VAE.



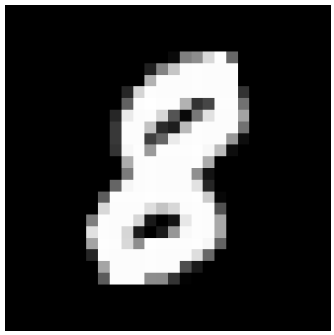
(k) Generation 10 VAE.



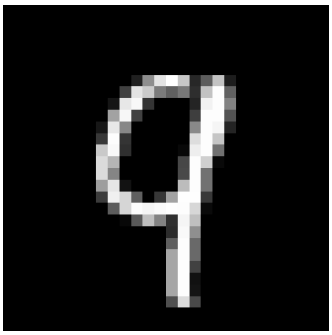
(l) Generation 20 VAE.



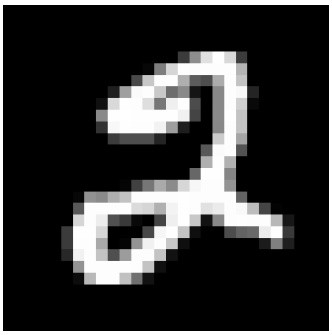
(m) Example MNIST image.



(n) Example MNIST image.



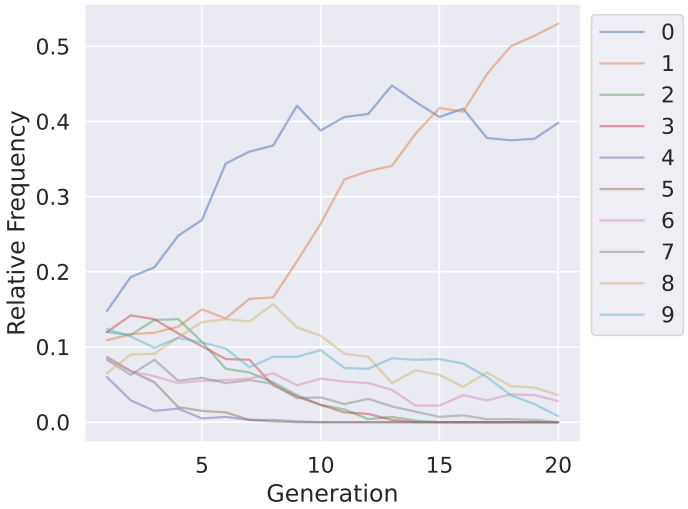
(o) Example MNIST image.



(p) Example MNIST image.

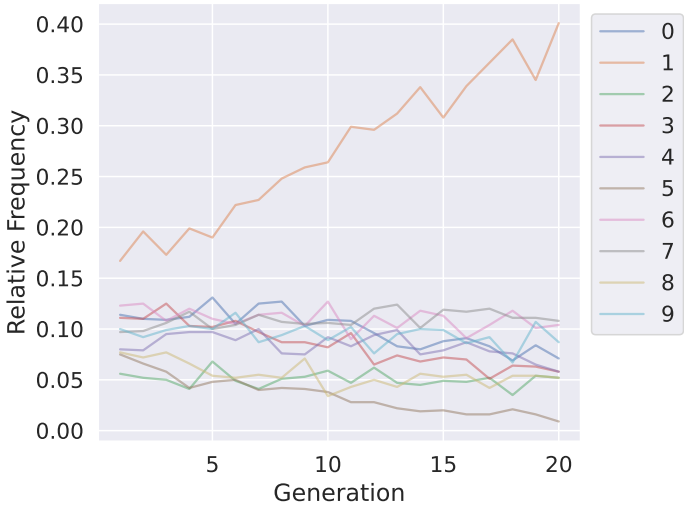
FIG. 5: We see clear indications of model collapse. (a-d): Images generated by generations 1, 5, 10, 20 of the WGAN model, using Setup 1. We see a retained variety of images generated by the later generations, though they have a decreased image quality. (e-h): We find a similar pattering using WGAN Setup 2. (i-l): Images generated by generations 1, 5, 10, 20 of the VAE model. (m-p): Example images from the MNIST dataset. The image intensities in the scaled MNIST dataset takes values in $[0, 1]$. Generated images with pixel intensity values outside this range are clipped, to ensure a similar scale for all images.

datasets. Just as the flattening of the images required to calculate the Wasserstein-1 distances in fig. 8(a) is an important caveat when interpreting the plots, so is the preprocessing of the images when interpreting the FID values. Again, due to the computational complexity and limited resources, we decided to calculate the distances using 1000 images for the different datasets. When extending and up-sampling the images from $1 \times 28 \times 28$ to $3 \times 299 \times 299$, the dimensionality increases, making the datasets quite sparse compared to the total space. How-



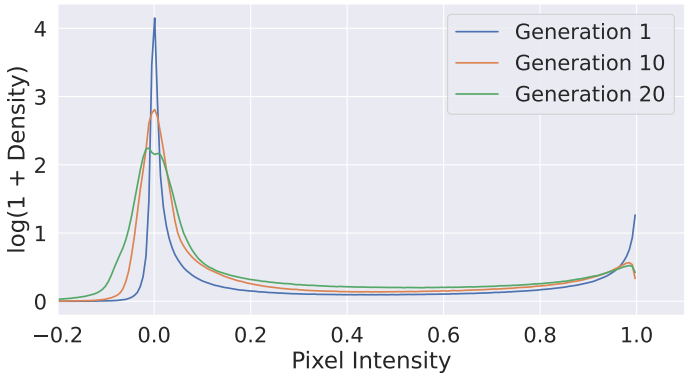
(a) VAE

(b) WGAN Setup 1

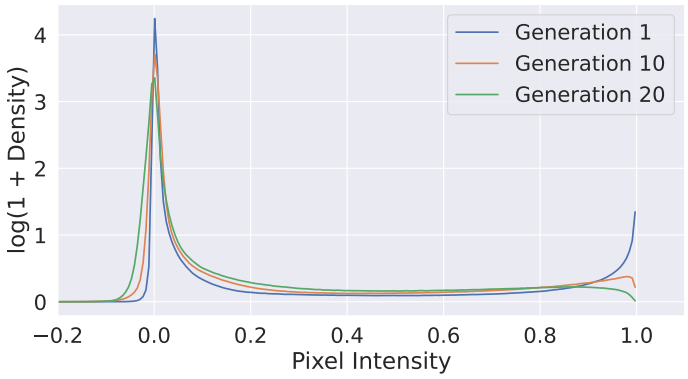


(c) WGAN Setup 2

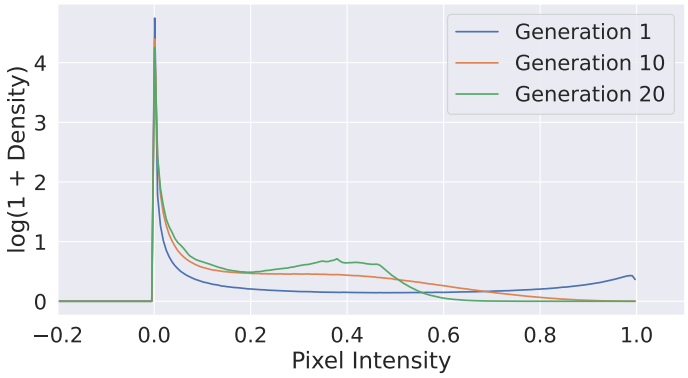
FIG. 6: Relative frequency of generated numbers, determined by a CNN classifier.



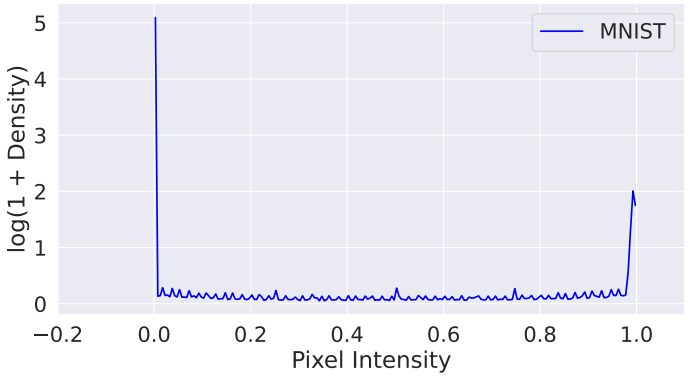
(a) WGAN Setup 1



(b) WGAN Setup 2



(c) VAE



(d) MNIST

FIG. 7: (a-c): Distribution of pixel intensity in images generated by different generations of the VAE decoder and WGAN generator. (d): The pixel distribution in the MNIST dataset, for comparison.

(a) Wasserstein-1 distance

(b) Fréchet inception distance

FIG. 8: (a): Wasserstein-1 distance between generated datasets of 500 flattened images and a random sample of 500 flattened MNIST-images, across generations and models. (b) FID between generated datasets of 1000 images and a random sample of 1000 MNIST-images, across generations and models.

ever, the content of the three channels is simply a duplication of the information contained in the original single channel. The up-sampling should not have too much an effect on the content within each channel. Therefore we would argue it reasonable to consider the distance metrics for the preprocessed datasets to be representative of that of the original datasets. From the plot we can tell that the general trend, as in fig. 8(a), is an increase in FID-values over generations for all models.

As noted, both the Wasserstein-1 distance and the FID increases with generation number, for all models. In addition, the ordering of the models seems to be consistent across the two metrics. Specifically, WGAN Setup 2 appears to be the most robust to iterated training, in terms of distributional distances. It is followed by the VAE, and WGAN Setup 1. This is an indication that WGAN Setup 2 might be the most favorable combination of model and training parameters to protect from model collapse.

We have observed a drift in data distributions that is consistent with model collapse as described in [6]. However, we want to expand on this point slightly. Though dataset distance appears to increase, it can do so in different ways. The clearest difference is in the variety of images encoded in the generated data distribution. The WGAN generators appear to retain a more robust multi-modality, while the images degrade in quality. This is particularly the case for WGAN Setup 1. We have slight indications of a drift towards mono-modality for WGAN Setup 2, as in fig. 6. Whereas the VAE clearly collapses to a single type of image. Hence we suggest the following updated terminology:

- *Model collapse* is a term reserved for a specific type of model degradation, for iterated training on generated data. Specifically, the type of model degradation where you observe a gradual collapse in mode, as is the case for the VAEs trained in this work.
- *Model drift* denotes model degradation due to iterated training, where the multi-modality of the data is maintained. This is more reminiscent of what we see in the the WGAN models, where the generated images are varied, but clearly suffer from lower quality in later generations.

Next we will consider some possible explanations for the VAE being more susceptible to model collapse. First of all, it could simply be a property of VAEs. Another possible explanation is that our specific VAE architecture is quite shallow compared to our WGAN architecture, as seen in figs. 3 and 4. It might be the case that the VAE is more robust, for architectures with more layers. It would also be interesting to test the efficacy of Convolutional VAEs, which make better use of the spatial structure in the image.

Finally, we remark that we have not been able to train more than 20 generations of models, due to computational constraints. This is a sufficient number of generations to observe model collapse for the VAE. A higher number of generations might very well lead to model collapse in the WGAN as well.

B. WGAN Discriminator Performance

Now we will examine the performance of the WGAN discriminators, and relate their performance to possible model collapse for their corresponding generators.

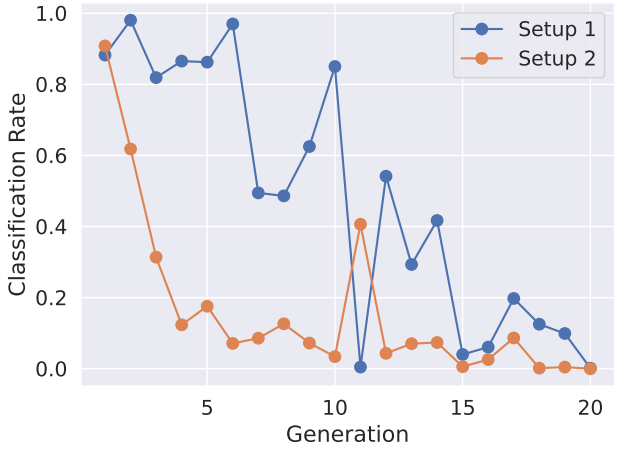


FIG. 9: The classification rate of the WGAN discriminators (using Setup 1 and Setup 2) on the MNIST test set, consisting of 10 000 images.

The WGAN discriminator classification rate on the MNIST test set is shown in fig. 9. It is clear that the classification rate is highest at generation 1 where it is around 90%. Also, Setup 1 performs better than Setup 2 for all generations except 1 and 10. Moreover, the classification rate is generally lower for later discriminator generations. This indicates that later discriminator generations "forget" the true distribution of images, found in the MNIST dataset. In turn, the generators trained against such discriminators will naturally produce images that are far away the original data distribution. This helps explain the increase in distributional distance for the WGAN-generated datasets shown in fig. 8.

We compute the classification rate of the discriminator from both WGAN setups, when applied to images generated by the VAE and both WGAN setups. The data is presented in heatmaps in fig. 10. There are clear diagonal patterns in subfigures c) and f). These patterns are due to the discriminator being paired to the generators from the same setup. Specifically, the first superdiagonal showcases lower classification rates. This makes sense, as the n'th generator creates the data that is used to train the $(n+1)$'st discriminator. Hence the discriminator will have a higher probability of judging that an image generated by the previous generation of generators is real. So this accounts for why we see a pattern of misclassification on the 1'st superdiagonal, but also for surrounding squares. For non-matching discriminator and generator setups we instead see a pattern of vertical lines where misclassification is more common. These lines are particularly distinct in subfigures a), b), and e). This pattern suggests that each discriminator performs similarly at classifying adjacent generations of generated images. Interestingly, WGAN Setup 2 appears to yield higher classifications rates for fake images, as compared to Setup 1.

Putting together the results in figs. 9 and 10, we see that WGAN Setup 1 is better at classifying real images and worse at classifying fake images, as compared to Setup 2. That is, WGAN Setup 1 has yielded models that tend to err on the side of predicting images to be real. On the other hand, Setup 2 resulted in models that with a higher frequency tended to label images as fake. It is not entirely clear if this actually is a result of the parameter choices in the setups. The effect might very well be explained by a random "direction of drift" that the models experience when trained recursively. That is to say, the effect could be explained by chance. Repeated experiments would be needed to map out if this is a real effect.

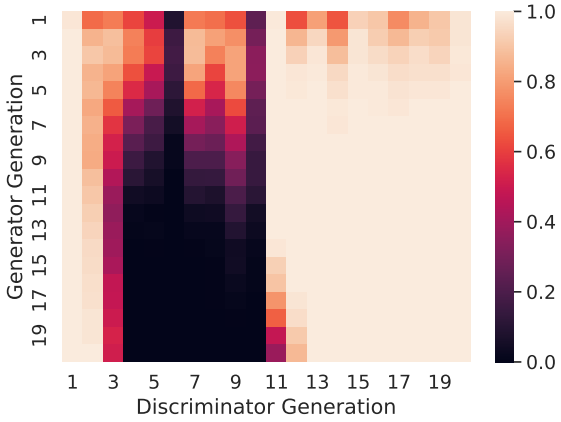
An interesting question is whether the discriminator or generator becomes "'bad" first. We see a gradual trend towards both performing worse and worse. And we see model degradation (albeit of a slightly different character) for both Setup 1 and Setup 2, the first of which emphasizes training the discriminator, and the second dedicating more iterations to training the generator. There is no obvious winner between the two setups in terms of discriminator performance. As pointed out, Setup 1 tends to skew in the direction of classifying images as real, and Setup 2 tends to classify more images as fake. Interestingly, Setup 2 seems to outperform Setup 1 in terms of generated image quality (at least by some metrics, see eg. fig. 8). This is a slight indication that Setup 2 outperforms Setup 1. And indeed this is a sensible result: Setup 2 sees the discriminator being trained $\frac{6}{5}$ times more than Setup 1. Furthermore, the Setup 2 generator has been trained 3 times more than the Setup 1 generator. This likely accounts for the discrepancy.

To nuance this point a bit further, it is not entirely obvious which number of critic iterations to use. In Setup 1, the discriminator is trained a lot more relative to the generator. And this is reasonable in the sense that you need to train a good discriminator first, before the generator can learn the distribution of images. However, the $5:1$ balance in training iterations may be too skewed. It has become apparent that the choice of this hyper-parameter can be quite influential on model quality. However, due to limited computational resources, we have not been able to explore this further.

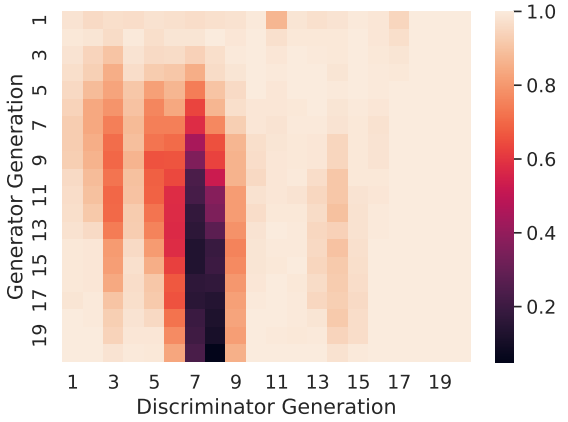
IV. Conclusion

This project has explored the performance and long-term reliability of two widely used generative modeling frameworks: WGANs and VAEs. In particular, the investigation centered on their behavior under recursive training conditions, wherein each model is trained iteratively on data generated by its previous iteration. Such conditions are becoming increasingly relevant as the internet contains more and more data generated by machine learning models.

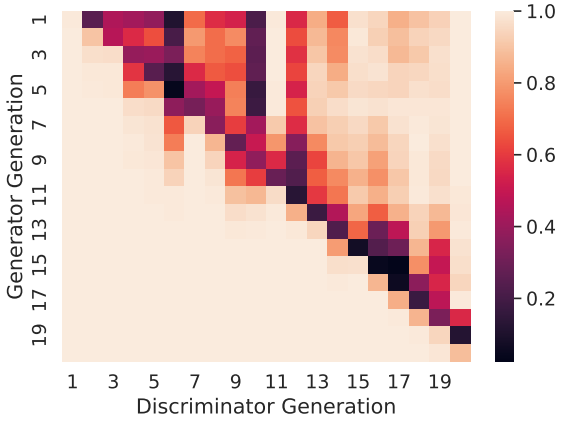
Empirical results revealed that both WGANs and VAEs



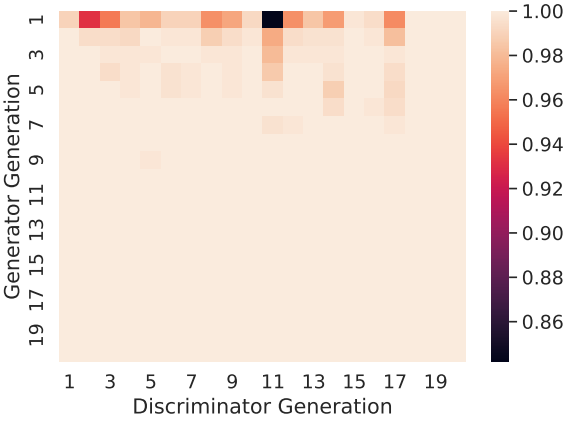
(a) Generator: VAE. Discriminator: WGAN Setup 1.



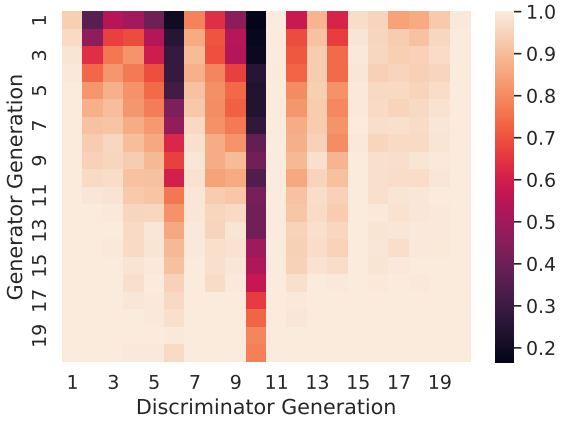
(b) Generator: VAE. Discriminator: WGAN Setup 2.



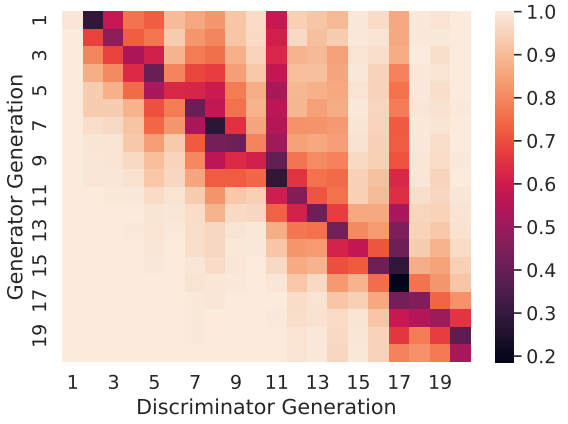
(c) Generator: WGAN Setup 1. Discriminator: WGAN Setup 1



(d) Generator: WGAN Setup 1. Discriminator: WGAN Setup 2



(e) Generator: WGAN Setup 2. Discriminator: WGAN Setup 1



(f) Generator: WGAN Setup 2. Discriminator: WGAN Setup 2

FIG. 10: (a, c, e): Classification rate of the WGAN Setup 1 discriminator on images generated by VAE, WGAN Setup 1, and WGAN Setup 2 respectively. (b, d, f): Classification rate of the WGAN Setup 2 discriminator on images generated by VAE, WGAN Setup 1, and WGAN Setup 2 respectively.

experience progressive degradation in output quality and diversity over generations, a process known as model collapse. This phenomenon was measured through a combination of visual analysis, image classification using a separately trained convolutional neural network, and distributional metrics including the Wasserstein-1 and Fréchet Inception distances. While early generations maintained fidelity to the source dataset, later generations deviated significantly, underscoring the fragility of generative stability in recursive scenarios.

The degree of degradation varied between model configurations, with certain WGAN setups demonstrating greater resilience. However, no architecture was immune to collapse, highlighting the limitations of current training paradigms. These findings indicate a need for further research into architectures, priors, and regularization strategies that can preserve performance over multiple generations of training.

In conclusion, while WGANs and VAEs remain powerful tools for data generation, ensuring their long-term viability requires a deeper understanding of collapse mechanisms and more rigorous evaluation methods. These considerations are essential as generative models are increasingly trained on synthetic data, sourced from the internet.

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[2] Youssef Kossale, Mohammed Airaj, and Aziz Darouichi. Mode collapse in generative adversarial networks: An overview. In *2022 8th International Conference on Optimization and Applications (ICOA)*, page 1–6. IEEE, October 2022.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN, 2017.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[6] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, July 2024.

[7] Xiaodan Xing, Fadong Shi, Jiahao Huang, Yinzhe Wu, Yang Nan, Sheng Zhang, Yingying Fang, Michael Roberts, Carola-Bibiane Schönlieb, Javier Del Ser, and Guang Yang. On the caveats of ai autophagy. *Nature Machine Intelligence*, 7(2):172–180, February 2025.

[8] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012.

[9] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-Learn*. Packt Publishing, Birmingham, England, February 2022.

[10] Cedric Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, Providence, RI, March 2003.

[11] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1–9, Beijing, China, 2014. JMLR Workshop and Conference Proceedings.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017.

[13] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.