Report : Demistifying MMD GANs

Ayoub Chenguel, Marlone Lequeux

This work is based on the paper 'Demistifying MMD GANs'

1 Introduction

Generative Adversial Networks (GANs) provides a general method to generate data following a complex distribution, like natural images, learning from real data.

The GAN architecture is made of two parts : a Generator that aims to generate data from noise (generally gaussian white noise), and a Discriminator that aims to discriminate between real and fake data. The discriminator classifies data according to the following rule : a high value means the discriminator classifies the data as coming from the original distribution. Discriminators and generators are usually parametrized by neural networks.
In the classic GAN framework, the discriminator's values are in $[0, 1]$, but we will see later that in other GANs, it will not be the case. The Generator is trained to trick the Discriminator into classifying generated images as coming from the training data distribution, and the Discriminator is trained to be as discriminative as possible. The classic GAN training attempts to solve the following min-max optimization problem (also called two players game):

$$\min_{G\in\mathcal{G}} \max_{D\in\mathcal{D}} \mathbb{E}_{Y\sim\mu_{\text{data}}}\left[\log D(Y)\right] + \mathbb{E}_{Z\sim\mu_Z}\left[\log\left(1 - D(G(Z))\right)\right]$$

where μ_{data} denotes the data distribution, and μ_Z is generally a gaussian isotropic random variable. By denoting $\mathcal{L}(D, G) = \mathbb{E}_{Y\sim p_{\text{data}}}[\log D(Y)] + \mathbb{E}_{Z\sim p_Z}[\log(1 - D(G(Z)))]$ If both G and the data have laws μ_G and μ_{data} of densities p_G and p_{data}, one can show that :

$$\sup_{D\in\mathcal{D}} \mathcal{L}(D, G) = -\log(4) + 2 \cdot \text{JS}(\mu_{\text{data}}, \mu_G)$$

Where JS denotes the Jensen-Shannon divergence defined by :

$$\text{JS}(\mu, \nu) = \frac{1}{2}\text{KL}\left(\mu \,\middle\|\, \frac{\mu+\nu}{2}\right) + \frac{1}{2}\text{KL}\left(\nu \,\middle\|\, \frac{\mu+\nu}{2}\right)$$

In the end, we see that by solving this two players game, the GAN aims to minimize a divergence between two probability measures. This idea is fundamental and will be used to make other formulation for GANs, that will rely on minimizing other metrics on probabilty spaces.

1.1 Limits of this formulation

First, if μ_{data} and μ_G have disjoint supports, ie if there is a measurable set A such that $\mu_{data}(A) = 1$ and $\mu_G(A) = 0$, the optimal discriminator is $D = \mathbb{1}_A$ And this does not depend on how close the supports of μ_{data} and μ_G are. This comes from the well known blindness of KL divergence to ground distances.

Second, when μ_{data} has finite support (which is the case during training, as we only have access to μ_{data} through data points), and μ_G has density, then , the optimal D is approximately 0 around fake points and 1 around data points. Thus, the gradient of the loss with respect to the generator's parameters is vainsihing almost everywhere : the generator does not learn anymore.

Other mathematical formulations have been explored to deal with this issue.

The most famous is the Wasserstein-GAN, which lies on the dual formulation of the 1-Wasserstein cost.

The optimization problem WGAN training aims solving is :

$$\inf_{G} \sup_{\|D\|_L\leq 1} \left[\mathbb{E}_{Y\sim\mu_{data}}[D(Y)] - \mathbb{E}_{Z\sim p_Z}[D(G(Z))]\right]$$

The main advantage of this formulation is that Wasserstein distance is sensitive to ground distance between the measures' supports, by construction of the Wasserstein distance. WGAN brings sensitiveness to ground distances to the GAN optimization problem, and the smoothness constraint on the discriminator function makes exploding gradients impossible. But this smoothness constraint must be tackled in the optimization. Different techniques arised. The first one is the weight-clipping : after every weight update during the training, weights are clipped into the interval $[-c, c]$, where c is a relatively small value. It ensures the discriminator respects the 1-Lipschitzness constraint. However this method limits the expressiveness of the discriminator, hence leading to poor performances compared to other techniques like gradient penalty. The gradient penalty technique consists in optimizing the following function :

$$\inf_G \sup_D \left\{ \mathbb{E}_{Y\sim\mu_{\text{data}}}[D(Y)] - \mathbb{E}_{Z\sim p_Z}[D(G(Z))] - \lambda\, \mathbb{E}_X \left[(\|\nabla_X D(X)\| - 1)^2 \right] \right\}$$

Hence constraining discriminator to be (almost) 1-Lipschitz. This method is way easier to train and gives notouriously better results than weight-clipping.

However WGAN suffer from a crucial problem : biased gradients. The WGAN training procedure produces biased gradients and may even converge to a wrong generator (bellemare et al 2017). This comes from the fact that it is impossible to find the perfect discriminator at each ascent step, and the paper proves that it produces a downard bias for the gradient. In this (bellemare et al 2017), MMD distance has been shown as an answer to this problem. However, we will further precise this statement. While MMD gradients are indeed unbiased, the MMD training procedure still produces biased gradients, but the MMD can be argued as 'less biased'. That is what is formulated in the following part.

2 IPM and MMD

Definition (Generalized IPM). Let $\mathcal{X}$ be some domain, with $\mathcal{M}$ a class of probability measures on $\mathcal{X}$. Let $\mathcal{F}$ be some parameter set, and let $J : \mathcal{F} \times \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ be an objective functional. The generalized IPM $D : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is then given by

$$D(P, Q) = \sup_{f\in\mathcal{F}} J(f, P, Q).$$

If $\mathcal{F}$ is a class of functions $f : \mathcal{X} \to \mathbb{R}$, and J is given by

$$J_{\text{IPM}}(f, P, Q) = \mathbb{E}_{X\sim P}[f(X)] - \mathbb{E}_{Y\sim Q}[f(Y)],$$

then we obtain **Integral probability metrics (IPM)**

Under existence, the **witness function** of an IPM is a function f^* such that $= J(f^*, P, Q) = \sup_{f\in\mathcal{F}} J(f, P, Q) = D(P, Q)$

Both MMD distance and 1-Wasserstein are integral probability metrics. Let's recall the construction of the MMD distance.

2.1 MMD Construction

MMD is an IPM for which the function class (denoted by $\mathcal{F}$) is the unit ball of a RKHS (Reproducing Kernel Hilbert Space).

Definition-Theorem (Aronszajn) Let $\mathcal{X}$ be a measurable space,

- K : $\mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is a positive kernel if $\forall (x_i)_{1\le i\le n}$, $(K(x_i, x_j))_{1\le i,j\le n}$ is a symetric positive matrix

For every positive kernel K, there exists a unique Hilbert space $\mathcal{H}$ such that : $\forall f \in \mathcal{H}, \quad f(x) = \langle f, k(x, \cdot)\rangle_{\mathcal{H}}$

This space $\mathcal{H}$ is called the reproducing kernel hilbert space (RKHS) of K

We define the **mean embedding** of the probability measure $\mathbb{P}$ as the element $\mu_{\mathbb{P}} \in \mathcal{H}$ such that $\mathbb{E}_{\mathbb{P}}[f(X)] = \langle f, \mu_{\mathbb{P}}\rangle_{\mathcal{H}} \quad \forall f \in \mathcal{H}$. It is given by $\mu_{\mathbb{P}} = \mathbb{E}_{X\sim\mathbb{P}}[k(\cdot, X)]$. The mean-embedding exists if $\mathbb{E}_{X\sim\mathbb{P}}[\|k(x, \cdot)\|_{\mathcal{H}}] < \infty$

The **maximum mean discrepancy (MMD)** is defined as the IPM with $\mathcal{F}$ being the unit ball in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$:

$$\text{MMD}(\mathbb{P}, \mathbb{Q}; \mathcal{H}) = \sup_{f\in\mathcal{H},\ \|f\|_{\mathcal{H}}\le 1} \left(\mathbb{E}_{X\sim\mathbb{P}}[f(X)] - \mathbb{E}_{Y\sim\mathbb{Q}}[f(Y)]\right).$$

If the Kernel veifies the previous integrability condition and if the application $\mathbb{P}` \to \mu_{\mathbb{P}}$ is injective (this is true if the kernel is characteristic, a theorical condition verified by usual kernels) then the MMD is a distance over probability measures on $(\mathcal{X}, \mathcal{M})$

And its witness function is proportionnal to $\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}$, i.e :

$$f^*(x) \propto \mathbb{E}_{X\sim\mathbb{P}}[k(X,x)] - \mathbb{E}_{Y\sim\mathbb{Q}}[k(Y,x)]$$

Exemple of kernels :

$$k_\sigma^{\text{rbf}}(x,y) = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$$

$$k_\alpha^{\text{rq}}(x,y) = \left(1 + \frac{\|x-y\|^2}{2\alpha}\right)^{-\alpha}$$

We also have the following

Theorem 1 : MMD Gradients are unbiased

$$\mathbb{E}_{X\sim P_m}\,\mathbb{E}_{Z\sim\mathcal{Z}^n}\,\partial_{\psi,\theta}\,\text{MMD}_u^2\left(h_\theta(X),\, h_\theta(G_\psi(Z))\right) = \partial_{\psi,\theta}\,\text{MMD}_u^2\left(h_\theta(P),\, h_\theta(G_\psi(Z))\right).$$

However we will see that when h is learnt adversially, gradients are still biased, as for the WGAN.

2.2 Energy Distance

The energy distance is an instance of the MMD. It is defined as the following :

$$D_e(P,Q) = -\frac{1}{2}\mathbb{E}_P[\rho(X,X')] - \frac{1}{2}\mathbb{E}_Q[\rho(Y,Y')] + \mathbb{E}_{P,Q}[\rho(X,Y)]$$

where ρ is a semimetric of negative type.

Sejdinovic et al. (2013, Lemma 12) showed that the energy distance is an instance of the maximum mean discrepancy, where the corresponding distance-induced kernel family for the distance is given by

$$k_{\rho,z_0}^{\text{dist}}(x,y) = \frac{1}{2}\left[\,\rho(x,z_0) + \rho(y,z_0) - \rho(x,y)\,\right].$$

2.3 MMD Particular Training

The fundamental difference between MMD and WGAN is that in the MMD case, the witness function is explicitely known, making the discriminator a priori non trainable. However, we do not expect that using classic kernels that does not depend on the data is a procedure that works. We will justify this later. The MMD training procedure is slightly different from GAN or WGAN training. In the MMD case, the generated points and the data points are passed in a CNN whose parameters are optimized. More precisely in the ascent steps, we try to solve the following problem :

$$\max_\theta \text{MMD}^2\big(h_\theta(\mathbb{P}),\, h_\theta(\mathbb{Q})\big)$$

The reason is the following. As the distribution we are trying to learn is complex and lies on high dimensionnal manifolds, we do not expect MMD based on classic kernels to capture the geometry of these data points. The kernel we use needs to 'adapt' to the data. To do this, we optimize a neural network to find the best representation of the data (i.e h_θ) so that the geometry is captured.

In practice, it is observed that training an MMD without modifying this objective function that "convolutional features are unstable and difficult to train". So, in practice, regularization techniques are used, which are the same that are used for WGAN training : weight clipping and gradient penalty. In our experiments, we used gradient penalty regularization. In our experiments, we observed that regularization was indeed necessary as without it, the optimization was stuck in local minimas.

3 Training procedure and Gradient Bias

The paper states that for Integral Probability Metrics, the classic training procedure produces biased gradients for the generator.

The training procedure is the following :

Consider the IPM distance , with objective J and parameter class $\mathcal{F}$. Suppose we observe i.i.d. samples $X \sim P^m$, $Y \sim Q^n$, for any two distributions P, Q. A data-splitting estimator is a function $\hat{D}(X, Y)$ which:

- randomly splits the sample X into X^{tr}, X^{te}, and Y into Y^{tr}, Y^{te},
- selects a critic function $\hat{f}_{X^{\text{tr}}, Y^{\text{tr}}} \in \mathcal{F}$, independently of X^{te}, Y^{te},
- and returns a result of the form

$$\hat{D}(X, Y) = \hat{J}\left(\hat{f}_{X^{\text{tr}}, Y^{\text{tr}}}, X^{\text{te}}, Y^{\text{te}}\right),$$

where $\hat{J}(f, X, Y)$ is an estimator of $J(f, P, Q)$.

The following theorems explain why it is really likely that IPM based gradients are biaised downards.

Theorem 2. Consider a data-splitting estimator of the generalized IPM D based on an unbiased estimator $\hat{J}$ of J: for any fixed $f \in \mathcal{F}$,

$$\mathbb{E}_{X \sim P^m,\, Y \sim Q^n}\left[\hat{J}(f, X, Y)\right] = J(f, P, Q).$$

Then either the selection procedure is almost surely perfect,

$$\Pr\left(J\left(\hat{f}_{X^{\text{tr}}, Y^{\text{tr}}}, P, Q\right) = D(P, Q)\right) = 1,$$

or else the estimator has a downward bias:

$$\mathbb{E}\left[\hat{D}(X, Y)\right] < D(P, Q).$$

Theorem 3 of the paper shows that there does not exist on a certain class of Probability measures :

Theorem 3. Let $\mathcal{P}$ be a class of distributions such that

$$\{(1 - \alpha)P_0 + \alpha P_1 : 0 \leq \alpha \leq 1\} \subseteq \mathcal{P},$$

where $P_0 \neq P_1$ are two fixed distributions. Let D be an IPM. Then, there does not exist any estimator of D which is unbiased on $\mathcal{P}$.

3.1 Gradient Bias for IPM

Let Q_ψ denote the distribution induced by a generator $G_\psi(Z) \sim Q_\psi$, where Z is a random variable (e.g. gaussian white noise) and ψ are the generator parameters. Then define:

$$D(\psi) := D(P, Q_\psi),$$

where D is an Integral Probability Metric between the true data distribution P and the model distribution Q_ψ.

The following theorem gives the final conclusion :

Theorem 4. Let $D : \Psi \to \mathbb{R}$ be a function defined on a parameter space $\Psi \subseteq \mathbb{R}^d$, with a random estimator $\hat{D} : \Psi \to \mathbb{R}$ which is almost surely differentiable. Suppose that $\hat{D}$ has unbiased gradients:

$$\mathbb{E}\left[\nabla_\psi \hat{D}(\psi)\right] = \nabla_\psi D(\psi).$$

Then, for each connected component of Ψ,

$$\mathbb{E}\left[\hat{D}(\psi)\right] = D(\psi) + \text{const}.$$

Theorem 2 and 3 indicates us a general framework where there is no unbiased estimator of an IPM distance. The previous theorem permits to conclude on the gradients : if an estimator of the IPM

distance has unbiased gradients, then the bias of the distance estimator does not depend on the generator parameters. As this is extremly unlikely to happen (with big neural networks, which are widely used in practice), we must conclude that gradients are very likely to be biased. Concretely, these bias come from the fact that, at each step, we need to maximize using data points and not knowing the true distribution, hence the true critic, we compute a biased estimate of the critic.

As Wasserstein-1 and MMD (and energy distance, as a particular case of MMD) are both IPM, this conclusion applies on both distances. For both, with a fixed dicriminator, gradients are not biased. However it is the training procedure that produces biased gradients.

But the paper gives reasons on why MMD seems 'less biased' than W1.

Let's consider the following example : If we have $\mathbb{P} = \delta_0$, and $\mathbb{Q}_{||} = \delta_q$, with $q > 0$. If we want to learn $\mathbb{P}$ with $\mathbb{Q}_{||}$

- If we use the Wasserstein-1 distance, the optimal distriminator will be given by $D(t) = t$. And if we hold this discriminator, and minimize over q, the optimal q will be $-\infty$
- However, if we use the MMD distance, the optimal distribution will always be $\mathbb{P}$

Thus, MMD unbiased gradients (gradients when the optimal discriminator is used) carry more information than the Wassertein ones, and in some sense, make the MMD 'less biased'. This might be because in the MMD case, the optimal witness function explicitly depends on the distributions we compute the distance on.

To draw a clear conclusion : the MMD Gradients are unbiased when used on top of a fixed representation, but are biased when the representation is learnt adversially. And the MMD-GAN efficiency comes from the fact that in practice, it requires smaller networks than WGAN to achieve the same performances, and that it can be argued as less biased than WGAN.

4 Kernel Inception Distance (KID) Evaluation

Quantitatively assessing the performance of GANs is a nontrivial challenge due to the inherently subjective nature of visual quality. While metrics such as the Inception Score (IS) and Fréchet Inception Distance (FID) are common, they suffer from significant limitations. Specifically, IS is sensitive primarily to class diversity and often neglects the target distribution entirely. FID, although more robust due to its reliance on the Wasserstein-2 distance between Gaussian approximations of deep feature distributions, can still yield biased estimates.

The Kernel Inception Distance (KID) described in the paper, by contrast, addresses these concerns effectively. KID uses the MMD metric with polynomial kernels. We hence have an unbiased estimator (of the squared MMD metric), eliminating the bias inherent in FID measurements. Furthermore, unlike the FID estimator, the KID estimator is asymptotically normal:

$$\sqrt{n}\left(\widehat{\mathrm{MMD}}^2 - \mathrm{MMD}^2(P, Q)\right) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \tag{1}$$

This enables statistical testing and robust performance comparisons. Consequently, the KID metric significantly reduces the risk of incorrect rankings between GAN models and facilitates more reliable decision-making in model selection processes.

4.1 Adaptive Learning Rate via KID-Based Statistical Testing

Optimizing GAN models typically involves tuning learning rate schedules manually, which can be inefficient and suboptimal. Inspired by practices in supervised learning, we propose an adaptive strategy for dynamically adjusting the learning rate based explicitly on statistical tests applied to the KID metric. This approach assesses whether model improvements are statistically significant over training iterations.

Specifically, it employs the relative similarity hypothesis test introduced by Bounliphone et al. (2016). This test evaluates the hypothesis , where denotes the generated distribution at iteration , and represents the validation distribution. Given that KID is an unbiased estimator of MMD, we directly use KID scores to instantiate this test. The procedure is as follows:

- Compute the KID scores for current and past iterations relative to a fixed validation set.
- Calculate the p-value of the test for improvement significance.

- If the p-value fails to reject the null hypothesis (no significant improvement) at a predefined significance threshold , this iteration is considered a failure.
- If the algorithm encounters a fixed number of consecutive failures, the learning rate is multiplicatively decreased, typically by a factor such as .

This adaptive mechanism systematically ensures that learning rate adjustments are justified by statistically robust evaluations, and allows to not have to set a threshold by hand.

5 Experiments

In this section, we detail the experimental evaluation of MMD GANs framework on different datasets. The goal is to empirically assess the capability of the model to capture the underlying distribution of data.

5.1 2D Dataset

We generate a synthetic 2D dataset using the `make_moons` function from `scikit-learn`, with $n = 1000$ samples and a noise level of 0.1. The inherent non-linearity and bimodal structure of the dataset make it a suitable benchmark for visualizing the performance of generative models.

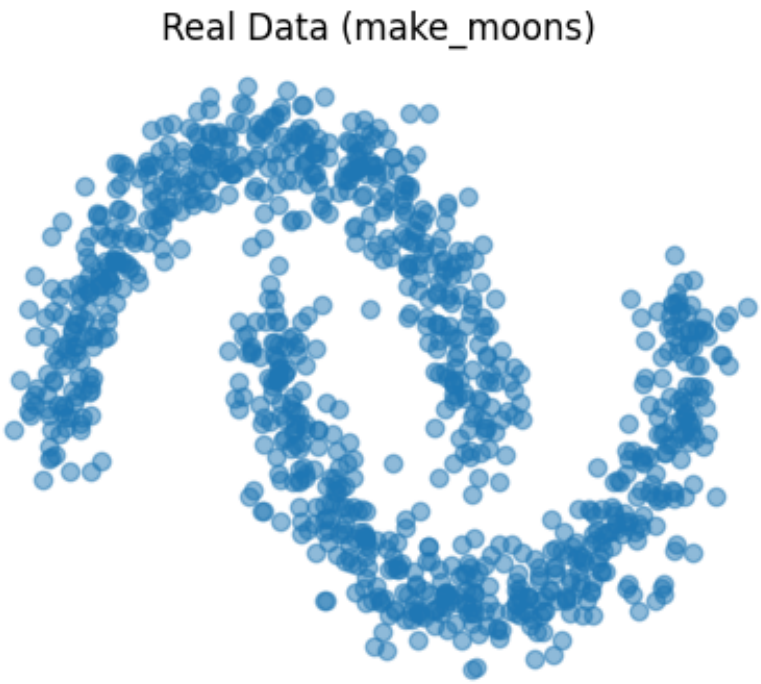

Figure 1: Make moon data points.

5.1.1 Model Architectures

Generator. The generator is implemented as a fully connected neural network with two hidden layers. Each hidden layer comprises 16 units and utilizes the ReLU activation function.

Critic. The critic network, also fully connected, is designed with an input layer, two hidden layers with 32 units each, and an output layer that projects the input into a 16-dimensional feature space. The critic serves both as a feature extractor and a discriminator.

It is important to note that the generator receives a noise vector of dimension 2. This low-dimensional latent space simplifies the mapping from noise to data, and it is particularly well-suited for our 2D

`make_moons` dataset. The choice of a 2D noise space facilitates direct visualization and intuitive interpretation of the model's performance, as we can clearly observe how the generator transforms the latent vectors into realistic samples.

5.1.2 Kernel Function

To compare the distributions of the real and generated data in the critic feature space, we employ a rational quadratic (rq) kernel. The kernel is defined as:

$$k_{\alpha}^{\mathrm{rq}}(x, y) = \sum_{\alpha \in \{0.2,1,5\}} \left(1 + \frac{\|x - y\|^2}{2\alpha}\right)^{-\alpha}$$

which effectively captures discrepancies at multiple scales. This multiscale approach ($\alpha \in \{0.2, 1, 5\}$) is crucial to account for both fine and coarse differences between the distributions. However, on this 2D dataset here it would work well with a single $\alpha = 1$.

5.1.3 Hyperparameters

The training procedure is governed by the following hyperparameters:

Parameter	Value	Description
Number of Epochs	10000	Total training iterations
Batch Size	128	Samples per mini-batch
Critic Iterations	5	Updates per generator update
Learning Rate	1×10^{-4}	For both generator and critic
Gradient Penalty Coefficient (λ_{GP})	1.0	Regularization strength

Table 1: Training hyperparameters.

The MMD loss defined as:

$$\mathcal{L}_C = -\mathrm{MMD}(f_{\mathrm{real}}, f_{\mathrm{fake}}) + \lambda_{GP}\, \mathcal{L}_{GP},$$

where $\mathcal{L}_{GP}$ is the gradient penalty. The generator, in contrast, minimizes the MMD:

$$\mathcal{L}_G = \mathrm{MMD}(f_{\mathrm{real}}, f_{\mathrm{fake}}).$$

5.1.4 Results

Visualization plays a pivotal role in assessing the performance of our MMD GAN. During training, we record:

- **Real vs. Generated Samples:** At periodic epochs, we plot both the real data and generated samples. The generated samples are overlaid on the real data to provide a visual comparison.
- **Critic Scalar Field:** We compute and visualize a scalar field by taking the L_2 norm of the critic's output over a grid covering the data domain. This field gives insight into how the critic differentiates between real and generated data.

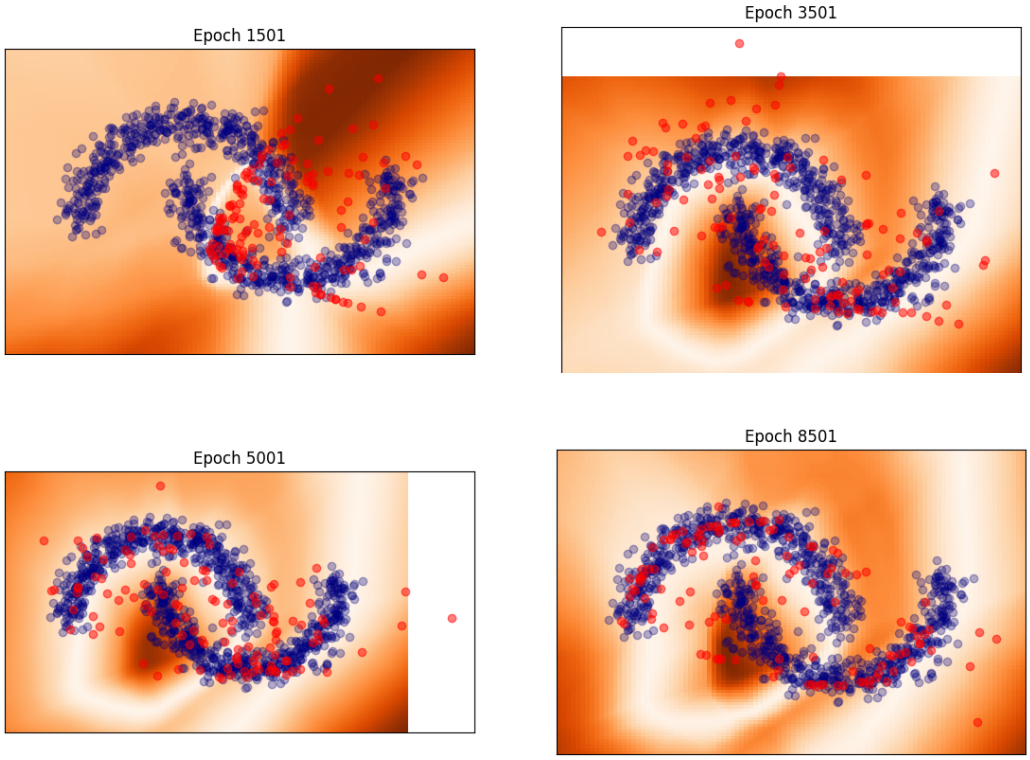

Figure 2: Visualization of the training progress. Each sub-image corresponds to a different stage of the training process.

After approximately three minutes of training, we visualize the final generations of our MMD GAN. As shown in the next figure, the generated samples align closely with the real data distribution, demonstrating the model's effectiveness in capturing the underlying structure of the dataset.

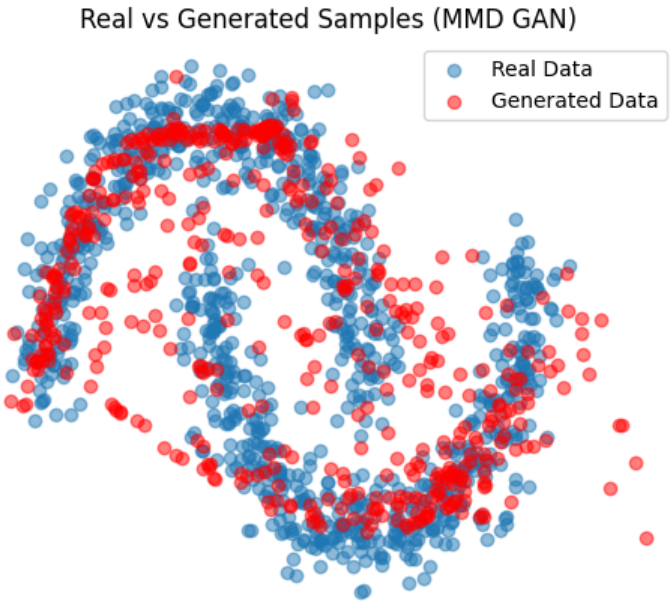

Figure 3: Final generated samples after 10 000 epochs

The evolution of the loss functions during training is monitored to assess convergence:

Figure 4: Training loss evolution for both networks.

On this simple dummy dataset we have a steady decrease of the loss:

- **Critic Loss:** The negative MMD loss, augmented with the gradient penalty, converges as the critic learns to maximize the discrepancy between the two distributions.
- **Generator Loss:** A decreasing MMD value indicates that the generator is effectively reducing the discrepancy.

5.1.5 Future Directions

The preliminary experiment on a synthetic 2D dataset was instrumental in acquainting us with the core components of our model, including the loss function, kernel design, and gradient penalty mechanism. With these insights, we are now poised to tackle a more challenging scenario by applying our approach to the Fashion MNIST dataset, as described in the following section.

5.2 Fashion MNIST

We evaluated our MMD-GAN using the Fashion MNIST dataset, which consists of 60,000 grayscale images of size 28×28 pixels, representing ten distinct categories of clothing items. Each image was normalized to the interval $[-1, 1]$ to align with the output range of the generator, employing a standard preprocessing.

Figure 5: Example real images from the Fashion MNIST training set.

5.2.1 Model Architectures

The generator was structured using a fully connected layer (mapping a 100-dimensional latent space into a $7 \times 7 \times 128$ feature tensor) followed by batch normalization and convolutional transpose layers. It maps a latent vector $\mathbf{z} \in \mathbb{R}^{100}$ drawn from a standard Gaussian distribution $\mathcal{N}(0, I)$ into a 28×28 image through nonlinear transformations and upsampling operations:

- **Fully-connected block**: transforms $\mathbf{z}$ into a $7 \times 7 \times 128$ feature map, followed by `BatchNorm1d` and ReLU.
- **Deconvolutional stack**: two `ConvTranspose2d` layers to upsample from 7×7 to 14×14 and finally 28×28, applying batch normalization and ReLU activations after each deconvolution, except for

the last layer which uses a tanh activation to keep pixel intensities in $[-1, 1]$.

The critic (discriminator) is a feature extractor with two strided convolution layers, each followed by a LeakyReLU activation. The second convolution is additionally followed by batch normalization. This produces a $128 \times 7 \times 7$ representation of the 28×28 input. The convolutional output is flattened and passed through a fully connected layer, ultimately mapping each image into a 1024-dimensional feature vector. These features are then used for the Maximum Mean Discrepancy (MMD) computation.

5.2.2 Kernel Selection

We experimented with several kernel functions for computing the MMD:

- **Gaussian (Exponentiated Quadratic) kernel**: a multi-scale version

$$k^{\text{rbf}}{}_{\sigma}(\mathbf{x}, \mathbf{y}) \;=\; \sum_{s \in \{2,5,10,20,40,80\}} \exp\!\Big(-\tfrac{\|\mathbf{x}-\mathbf{y}\|^2}{2s^2}\Big).$$

- **Rational Quadratic (rq) kernel**:

$$k_{\alpha}^{\text{rq}}(x, y) = \sum_{\alpha \in \{0.1,0.2,1,2,5\}} \left(1 + \frac{\|x - y\|^2}{2\alpha}\right)^{-\alpha}$$

- **Composite kernel (rq*)**: a sum of the Rational Quadratic kernel and a linear (dot product) term,

$$k_{\alpha}^{\text{rq}^*}(x, y) \;=\; k_{\alpha}^{\text{rq}}(x, y) \;+\; \langle x,\, y \rangle.$$

Following the indications in the paper, we tried these three kernels.

5.2.3 Hyperparameters

Hyperparameter tuning was extensively performed, particularly focusing on the gradient penalty coefficient λ_{gp}. We explored three settings: $\lambda_{\text{gp}} \in \{0, 1, 10\}$. Empirical observations indicated that a moderate gradient penalty of $\lambda_{\text{gp}} = 1$ typically stabilized training more effectively than no penalty ($\lambda_{\text{gp}} = 0$) or higher penalties (e.g., $\lambda_{\text{gp}} = 10$).

We used a batch size of 128, the Adam optimizer with learning rates of 1×10^{-4} for both the generator and the critic, and $\beta_1 = 0.5$, $\beta_2 = 0.999$. The critic was updated 5 times for each generator update.

5.2.4 Results

The experimental results underscore critical insights regarding kernel choice and gradient penalty efficacy. Figure 6 demonstrates that using a Gaussian kernel with $\lambda_{\text{gp}} = 10$ yielded poor-quality outputs, characterized by slow convergence and unstable training behavior. Conversely, significant qualitative improvements were observed upon reducing the gradient penalty to $\lambda_{\text{gp}} = 1$ and employing the Rational Quadratic kernel, as illustrated in Figure 7.

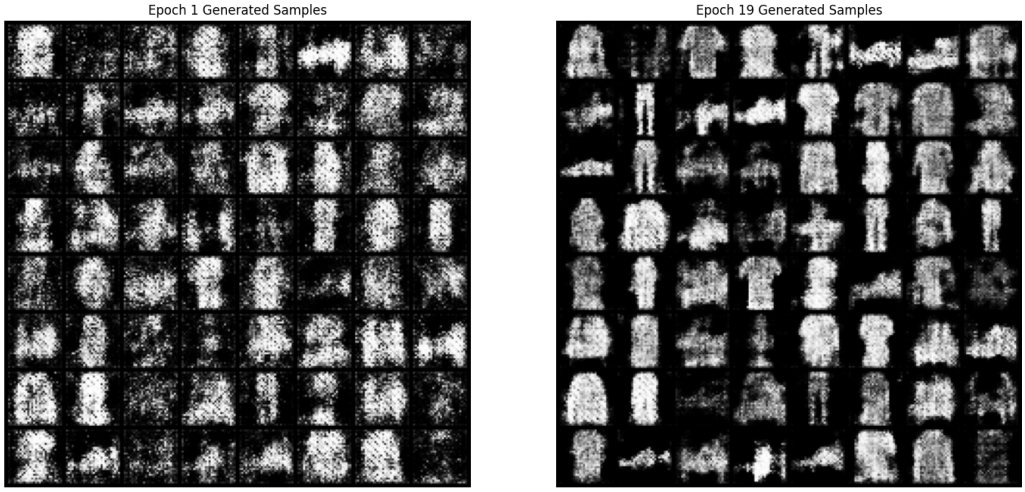

Figure 6: Generated images using the Gaussian kernel with $\lambda_{\text{gp}} = 10$: poor convergence and bad samples.

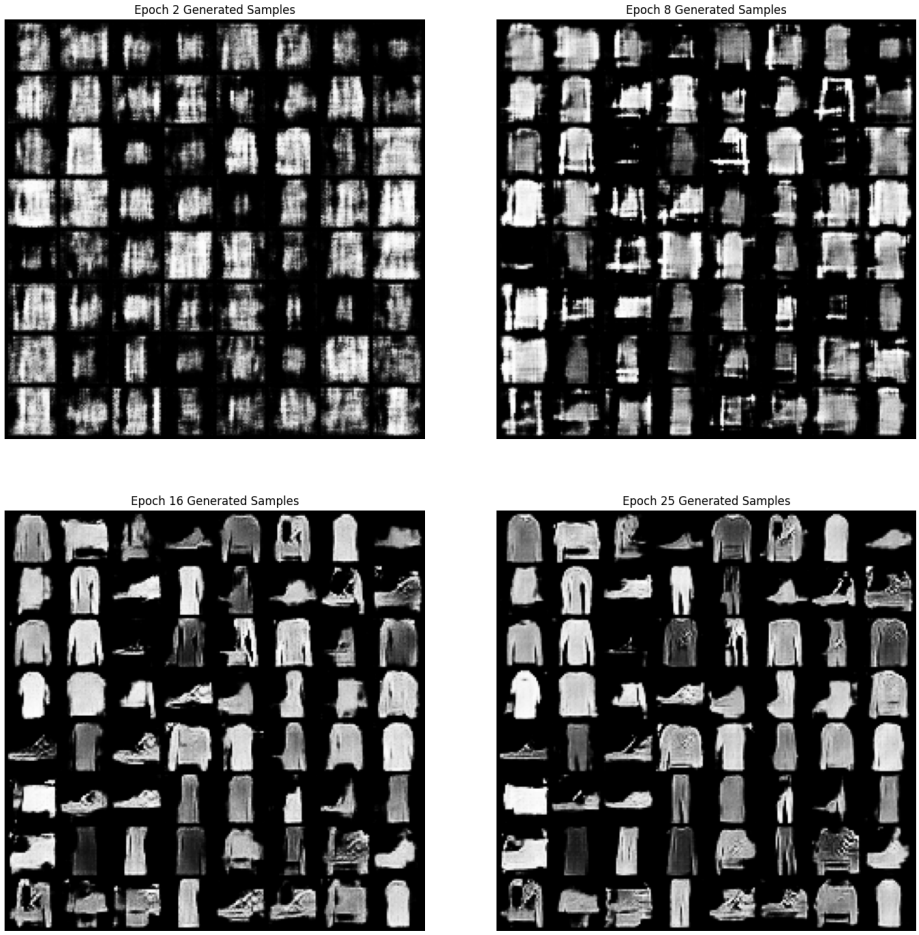

Figure 7: Improved training results using the Rational Quadratic kernel and $\lambda_{\text{gp}} = 1$. Samples show more coherent structures but still exhibit occasional artifacts.

The combination of Rational Quadratic kernel and moderate gradient penalty ($\lambda_{\text{gp}} = 1$) resulted in visually more appealing images, significantly closer to the target distribution as training progressed. Nevertheless, as shown in Figure 7, some artifacts remain, particularly on items such as shoes, indicating room for further refinement.

Additionally, omitting the gradient penalty entirely ($\lambda_{\text{gp}} = 0$) led to instability and degraded convergence, as displayed in Figure 8. This provides evidence that enforcing Lipschitz-like constraints is critical for preventing mode collapse when training MMD-based GANs.

Figure 8: Training instability observed when omitting the gradient penalty ($\lambda_{\text{gp}} = 0$).

Overall, these results clearly demonstrate the practical utility and necessity of a properly tuned gradient penalty, confirming its empirical effectiveness in conjunction with an MMD objective. Our experiments highlight that:

- The **Rational Quadratic kernel** generally performed better than the pure Gaussian kernel for Fashion MNIST.
- A moderate gradient penalty (e.g., $\lambda_{\text{gp}} = 1$) is pivotal for stabilization; too large a penalty (e.g., $\lambda_{\text{gp}} = 10$) can hinder convergence.
- Artifacts in certain classes, such as shoes, could probably have been avoided with further training.