Improving GAN Precision and Recall through GMM and Gradient Ascent

Groupe **DSL**

Louis Carron
Sacha Khosrowshahi
Ayouba Deba

IASD

1 Introduction

Generative Adversarial Networks (GANs) are known to exhibit a fundamental trade-off between precision and recall. High precision reflects the ability to generate sharp and realistic samples, whereas high recall indicates that the model captures the full diversity of the data distribution. In practice, standard GANs struggle to balance these two objectives: improving the sharpness of generated samples often leads to mode collapse, while encouraging diversity can degrade visual quality.

In this project, we investigate how the structure of the latent space and the training dynamics can be modified to improve this trade-off. Using the MNIST handwritten digits dataset, our group focused on two complementary approaches: Gaussian Mixture Models (GMMs) to impose a multi-modal structure in the latent space, and Gradient Ascent techniques to explicitly promote diversity or quality during generation. We also explored how these two strategies can be combined to further improve the precision–recall balance.

2 Model Improvement methods

2.1 Gaussian Mixture Models

Intuitively, real-world data does not occupy the entire high-dimensional space uniformly. Instead, it forms regions of low and high density, where samples within the same dense region tend to share similar patterns or semantic features. Therefore, the underlying data distribution we try to learn in the latent space may follow the same principle.

Building on this idea, we implemented the Gaussian Mixture GAN (GM-GAN [2]) model. In this framework, the latent space is no longer sampled from a single isotropic standard Gaussian, but from a mixture of K Gaussian components, or clusters:

$$z \sim \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k),$$

where π_k are the mixture weights (set uniform in our study $\pi_k = \frac{1}{K}$), $\mu_k \in \mathbb{R}^d$ the component means, and $\Sigma_k \in \mathbb{R}^{d\times d}$ their covariance matrices, with d the dimension of the latent space. This structure encourages the generator to map different latent regions to semantic modes of the data. In the MNIST case, with $K = 10$, it will learn to map the different digit to a cluster, allowing afterward a feature-wise control of the diversity and quality trade-off. We assumed the covariance matrix to be diagonal and given by the parameter σ_k such as $\Sigma_k = \sigma_k I$. It constraints the cluster by imposing independance between the latent dimensions, simplyfing optimisation but reducing its flexibility, meaning its freedom to shape its distribution during learning. It will allow us, depending on the context, to tune the value of σ to observe a desired trade-off between quality and diversity in Gaussian clusters.

We chosed to implement one variant of the original mixture model, differing in how the Gaussian parameters are defined or learned. Indeed, we allow **learnable** Gaussian means and a **fixed** isotropic covariance matrix $\Sigma_k = \sigma I$ in the training process. This configuration keeps the same dispersion for all components, while allowing the mixture to reorganize itself by adjusting the position of each Gaussian center.

2.2 Discriminator gradient flow

Having introduced classical refinement techniques based on Gaussian mixture models, we now try to implement a more general and principled framework based on variational calculus: *Discriminator Gradient Flow* (DGFlow) [1]. The core idea is to refine generated samples by simulating the gradient flow of an entropy–regularized f-divergence between the model distribution ρ and the target data distribution μ. More precisely, DGFlow defines the following functional:

$$\mathcal{F}_{f,\mu}(\rho) = D_f(\mu \,\|\, \rho) - \gamma H(\rho) = \int f\left(\frac{\rho(x)}{\mu(x)}\right) \mu(x)\, dx + \gamma \int \rho(x) \log \rho(x)\, dx, \tag{1}$$

where f is a convex function defining an f-divergence, and the entropy term $H(\rho)$ encourages sample diversity during finite-time evolution.

We clearly understand that the goal is to have an energy-based optimization on latent vectors in order to have a "pre-process" to approach the goal distribution and still have diversity in the latent vectors before feeding them to the generator. Since the true density ratio ρ/μ is unknown, DGFlow uses the discriminator as a density ratio estimator. Given a trained binary classifier $D(x) = \Pr(\text{real} \mid x)$, the density ratio can be recovered via:

$$\frac{\rho(x)}{\mu(x)} = \frac{1 - D(x)}{D(x)} = e^{-d(x)}, \tag{2}$$

where $d(x)$ is the discriminator logit. Finally, instead of updating samples directly in pixel space, DGFlow is applied in the latent space z of the generator $x = g_\theta(z)$. The final refinement update is given by :

$$z_{n+1} = z_n - \eta \nabla_z f'\big(e^{-d(g_\theta(z_n))}\big) + \sqrt{2\gamma\eta}\,\xi_n, \tag{3}$$

The hyper-parameters that are going to influence the behavior of the refinements are η, the lower it's going to be the less our model will allow itself to loop in "known" regions since we're giving less weight to the gradient. Moreover another parameter that is going to play a consequent role in the Recall is γ since it gives control on the intensity of the noise which will allow, or not, diversity in our samples.

3 Experimental results

3.1 Metrics

To compute **precision** and **recall**, we first extract feature embeddings by passing all images through the CNN and removing the final fully connected layers. In this feature space, each sample is assigned a local density radius defined by the Euclidean distance to its k-th nearest neighbour. Precision is then measured as the proportion of generated samples that fall within the density radius of at least one real sample, while recall corresponds to the proportion of real samples that are similarly covered by at least one generated point. This k-NN formulation provides a simple but effective geometric approximation of how well the generator matches both the quality and the diversity of the real data distribution. We set $k = 3$ as it provides a good empirical balance between strict local neighbourhood estimation and metric stability, avoiding the over-sensitivity of very small k and the overly permissive radii obtained with larger values.

For FID, we compare the distributions of real and generated samples in the CNN feature space. We approximate both distributions by multivariate Gaussians, estimating their means and covariances, and compute the Wasserstein-2 distance between them.

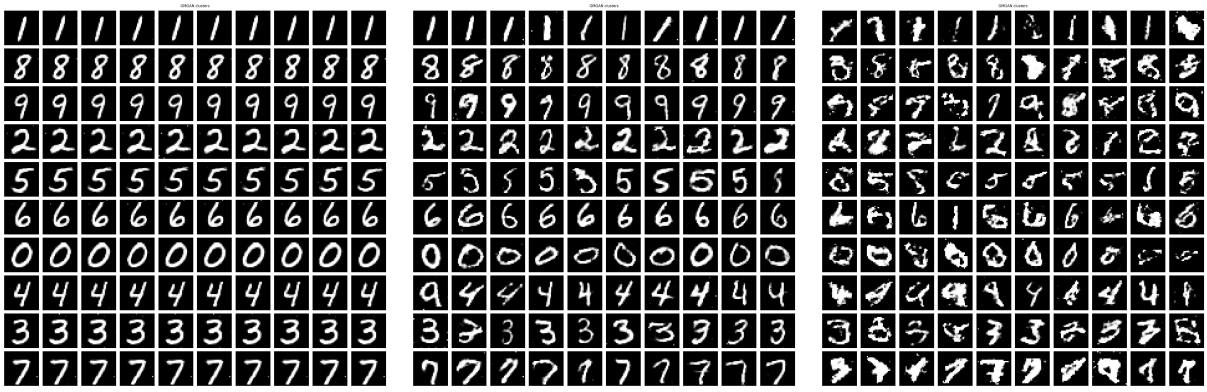
$$\mathcal{N}(\mu_r, \Sigma_r) \quad \text{and} \quad \mathcal{N}(\mu_g, \Sigma_g),$$

$$\text{FID}\big(\mathcal{N}(\mu_r, \Sigma_r),\, \mathcal{N}(\mu_g, \Sigma_g)\big) = \|\mu_r - \mu_g\|^2 + \text{Tr}\Big(\Sigma_r + \Sigma_g - 2\,(\Sigma_r \Sigma_g)^{1/2}\Big)\,.$$

This Gaussian assumption, significantly simplifies the computation by reducing the comparison of two complex feature distributions to closed-form operations on their first two moments.

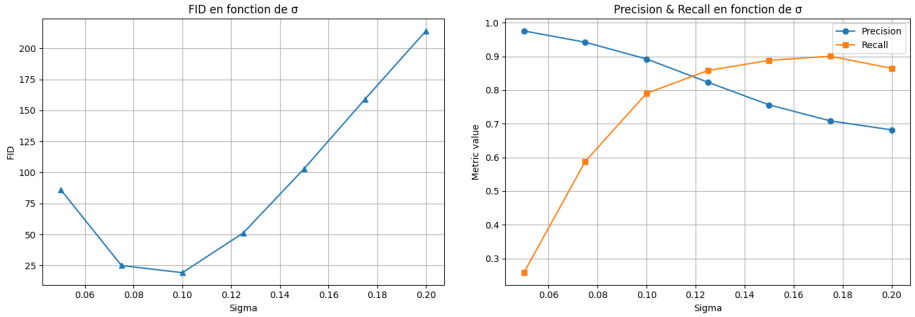
3.2 Gaussian Mixture Models

The GM-GAN model was trained using the original loss function of the Goodfellow article [3]. Instead of sampling from a standard distribution, it picks a cluster uniformly and draw samples for the selected mixture, then performs a learning step. To avoid collapse in the first few iterations, we add regularization in the discriminator training, with a dropout layer ($p = 3$) and gaussian noise on the real images. We also used to different value of batch for the generator and the discrimator, $b_G = 128$ and $b_D = 64$, and update the weights using Adam optimizer. We chose the number of cluster the same as the number of class in the MNIST dataset, so $K = 10$. The cluster means are initialized by choosing values uniformly at random in the range $[-c, c]$ for each latent dimension. We picked the same value as in the article, $c = 0.1$.



(a) $\sigma = 0.01$ (b) $\sigma = 0.1$ (c) $\sigma = 0.2$

Figure 1: Generated samples across clusters for three values of σ



(a) FID as a function of σ (b) Precision and Recall as a function of σ

Figure 2: FID and Precision–Recall curves across different values of σ.

This quantitative behaviour is consistent with the qualitative inspection of generated samples given Figure 1, where the cluster-wise generations remain coherent across multiple values of σ, further confirming the stability of the training. We observe that each cluster consistently maps to a specific digit class, showing that the GMGAN successfully learns to separate the data modes. The parameter σ controls the precision–recall trade-off: small values produce tighter clusters and higher precision, while larger values expand the cluster support, increasing diversity at the cost of visual quality.

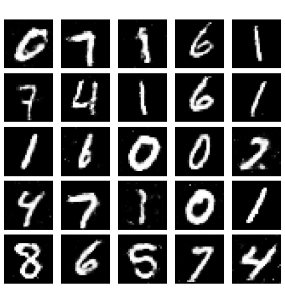
Figure 2 confirms our previous qualitative observations on the precision–recall trade-off: the FID curve exhibits a clear elbow with a minimum at $\sigma = 0.1$, indicating the best compromise between precision and diversity, achieved at $\sigma = 0.1$ with Precision $= 0.89$, Recall $= 0.79$, and FID $= 19.19$.

3.3 Gradient Ascent

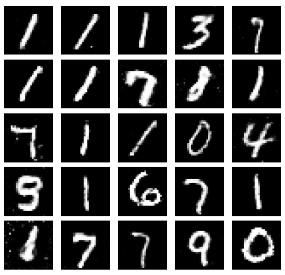
We are trying in this method to manipulate the trade-off between the entropy part of the f-divergence that influences the diversity and therefore Recall of our samples, and the divergence between the goal distribution and our sample distribution which has direct impact on the Precision of our samples.

In practice, this balance is controlled by three hyperparameters:

- **Step size** η
  Controls the magnitude of the gradient ascent step applied to the latent variables. Larger values of η lead to more aggressive refinements, increasing fidelity (Precision) but risking mode collapse and reducing diversity (Recall). Smaller values of η produce smoother changes and better preserve the initial latent structure. We can see below that there is collapse (the "easier" modes for the model to generate being 1, 7 and 9)

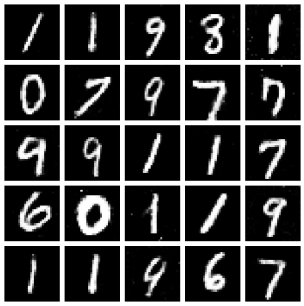


Low η (stable, diverse)

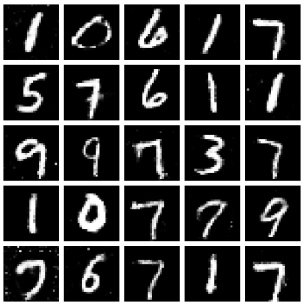


High η (sharper, less diverse)

- **Diffusion coefficient** γ
  Scales the stochastic noise injected at each refinement step. A higher γ increases entropy and prevents collapse, improving Recall at the cost of Precision. When $\gamma = 0$, the refinement becomes fully deterministic and strongly mode-seeking.

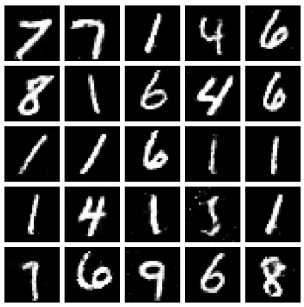


$\gamma = 0.1$ (deterministic)

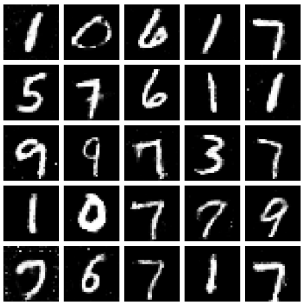


$\gamma = 0.4$ (more diverse, noisy)

- **Number of refinement steps** t
  Determines how long the refinement process is applied. Increasing t pushes samples further toward high-density regions identified by the discriminator, improving alignment with the real distribution (higher Precision). However, excessively large t can lead to over-refinement, reduced diversity, and degradation of Recall.

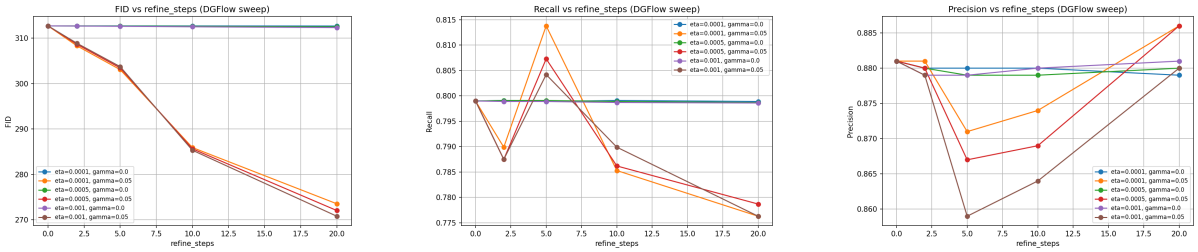


$t = 5$ steps



$t = 20$ steps (more refined but less diverse)

Finally, we need to compare these hyper-parameters values on Recall, Precision and FID :



FID Recall Precision

Figure 3: Comparison of hyper-parameters on FID, Recall and Precision.

We can see in all of these plots that when $\gamma = 0$ the results are always bad, this makes sense since we are not using the entropy term in our refinement. Another point that we need to assess is the high FID we are getting on visually acceptable results. The problem seems to be in the way we computed the FID when trying this gradient flow method. Computing the necessary statistics on the latent vectors may have been done the wrong way. Regardless, the best combination for Recall, Precision and FID seems to be the triplet $(\eta = 5 \times 10^{-4}, \gamma = 5 \times 10^{-2}, t = 5)$

4 Conclusion

By organizing the latent space into separate Gaussian clusters, the model naturally assigns each cluster to a digit class, which prevents mode collapse and preserves diversity. The Gaussian Mixture Model therefore gives us a way to impose structure on the latent space, making class separation explicit. Discriminator Gradient Flow, on the other hand, approaches the problem from the opposite direction: instead of structuring the latent space *before* generation, it refines already generated samples *after* generation by transporting them toward regions of higher discriminator density. This method explicitly optimizes the fidelity of individual samples, and the hyperparameters (η, γ, t) allow us to continuously move along the Precision–Recall frontier.

Taken together, these two techniques address complementary parts of the generative process. The GMM provides a global, interpretable prior that enforces diversity and prevents mode collapse at sampling time, while the refinement dynamics offer a local, sample-wise correction mechanism that can recover high-quality samples without retraining the generator. In principle, the two methods could be combined: one could first draw z from a structured GMM prior to guarantee semantic coverage, and then apply a small number of discriminator-driven refinement steps to improve fidelity while maintaining class balance and diversity. Such a hybrid approach would allow us to decouple global structure (handled by the GMM) from local quality control (handled by gradient flow), and may offer a more robust alternative to the traditional GAN paradigm, where both objectives must be learned simultaneously and often conflict. Exploring this integration represents a promising direction for future work, especially in problems where both interpretability and high visual quality are required.

References

[1] A. F. Ansari, M. L. Ang, and H. Soh. Refining Deep Generative Models via Discriminator Gradient Flow. In *ICLR*, 2021.

[2] M. Ben-Yosef and D. Weinshall, *Gaussian Mixture Generative Adversarial Networks for Diverse Datasets, and the Unsupervised Clustering of Images*, arXiv:1808.10356, 2018.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Nets*, arXiv:1406.2661, 2014.