Self-Diagnosing GANs

Sarp Tan Geçim (4175173) — Sevde Yanık (4732565)

`github.com/Sarptan/DiaGAN-Project`

Generative Neural Networks for Sciences

30.03.2025

Abstract

In this project, we replicate and evaluate the Self-Diagnosing Generative Adversarial Network (Dia-GAN) [LKHC21], a framework designed to detect and enhance the representation of under-represented data samples during GAN training. Traditional GANs often suffer from mode collapse, where the generator fails to capture the full diversity of the data distribution, particularly ignoring rare or minority patterns. Dia-GAN addresses this limitation by leveraging internal discriminator statistics—specifically, the Log-Density Ratio Mean (LDRM) and Log-Density Ratio Variance (LDRV)—to quantify which real samples are poorly modeled by the generator. These statistics are used to guide score-based weighted sampling, ensuring that hard-to-learn samples receive more focus during training.

Our implementation includes baseline and Dia-GAN variants across multiple datasets, including CIFAR-10, Colored MNIST, and a custom Gaussian Mixture toy dataset. We further investigate the use of Discriminator Rejection Sampling (DRS) as a post-processing step to enhance generation quality. Evaluation is carried out using Frechet Inception Distance (FID), as well as precision and recall metrics computed on classifier embeddings. Results demonstrate that Dia-GAN improves mode coverage and sample diversity compared to standard GAN baselines, with minimal compromise in fidelity. Our findings also highlight practical considerations when deploying Dia-GAN in resource-constrained environments.

Contents

1 Introduction [Gecim]

Generative Adversarial Networks (GANs) have shown remarkable results in various generative tasks. These include image synthesis, data augmentation, style transfer, domain adaptation, and more. Their ability to learn complex, high-dimensional data distributions has made them a popular choice in many fields. However, despite their success, GANs are known to suffer from a common and problematic issue called mode collapse. This happens when the generator learns to produce only a limited subset of the data distribution, effectively ignoring other regions or modes. In simple terms, the generator "figures out" how to fool the discriminator with samples from only the easier or more frequent parts of the dataset, and fails to capture the full diversity of the real data.

This is especially problematic when working with datasets that contain underrepresented or rare samples — for example, minority classes in classification datasets or fine-grained visual features in image datasets. These parts of the distribution are often crucial in real-world applications, and missing them reduces the usefulness and fairness of the generated data. Mode collapse leads to generators that look good at a glance but perform poorly in terms of diversity, limiting their application in areas like unbiased content generation or fair synthetic data creation.

To address this issue, several techniques have been proposed over the years. Some aim to modify the latent space, for example by truncating it or changing how samples are drawn. Others introduce post-training corrections, like rejection sampling, to discard less convincing outputs. While these strategies can help to a certain degree, they usually operate outside the core training process and don't directly help the model learn the missing parts of the data distribution.

In this project, we focus on the Self-Diagnosing GAN (Dia-GAN), which takes a different approach. Instead of relying on external fixes, Dia-GAN incorporates a score-based weighted sampling strategy directly into the training process. The main idea is to use statistics from the discriminator to track which real samples are persistently underrepresented during training. By computing the Log-Density-Ratio Mean (LDRM) and Log-Density-Ratio Variance (LDRV) over time, Dia-GAN assigns higher sampling probabilities to "hard-to-learn" examples. This way, the model puts more focus on data it would otherwise ignore, potentially improving both diversity and coverage without significantly changing the architecture.

In our project, we aim to replicate this idea and test its effectiveness in practice. Specifically, we ask the following research questions: Are the mean and variance of the Log-Density Ratio good indicators for identifying underrepresented samples during GAN training? Does using these statistics for weighted sampling actually help the generator learn more diverse outputs? And how does Dia-GAN compare to conventional GANs and other mode-collapse-aware techniques in terms of quality and diversity? We are also interested in the practical side: how sensitive is this method to different hyperparameters, and how well does it work with limited computational resources?

The main contributions of this work are:

- Implementation of the Dia-GAN framework using simplified but effective GAN architectures (CNNGAN, SNGAN, SSGAN)
- Empirical evaluation of LDRM and LDRV metrics in identifying underrepresented samples
- Comparison with baseline GAN models and additional sampling strategies like DRS
- Evaluation of trade-offs between fidelity and diversity using relevant metrics

While the overall idea behind Dia-GAN is promising, we expect several challenges along the way. Tracking LDRM and LDRV values over many epochs introduces extra computational and memory overhead. Also, the performance of the model might be quite sensitive to tuning parameters like the weighting factor k and clipping thresholds used in sampling. The results might also vary depending on the dataset, especially when the degree of imbalance or complexity differs. We hope to explore these aspects in our experiments and provide insights into both the strengths and limitations of the Dia-GAN framework.

2 Background [Gecim]

2.1 Fundamentals of Generative Adversarial Nets

Generative Adversarial Networks (GANs) are a class of deep learning models introduced by Goodfellow [GPAM+14] in 2014. They are designed to learn and generate new samples that closely mimic a target data distribution. GANs consist of two neural networks — a generator G and a discriminator D — that are trained simultaneously in a two-player minimax game. The generator takes as input a random noise vector z, sampled from a known prior distribution p_z (usually Gaussian or uniform), and maps it to the data space in an attempt to produce realistic-looking outputs. The discriminator, on the other hand, receives both real samples from the dataset and generated samples from the generator, and tries to correctly classify them as real or fake.

The generator's objective is to generate samples that are indistinguishable from real data, essentially trying to "fool" the discriminator. The discriminator's goal is to correctly identify whether an input sample is real or generated. This adversarial setup can be formulated as a minimax optimization problem, with the following objective function:

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

In theory, this game reaches an equilibrium when the generator produces samples that perfectly match the real data distribution, and the discriminator becomes maximally confused — outputting 0.5 for every input. At that point, the generator has successfully learned to replicate the true data distribution p_{data}.

However, in practice, achieving this equilibrium is extremely challenging. GANs are notoriously hard to train due to several factors: sensitivity to hyperparameters, unstable dynamics between the generator and discriminator, and gradient vanishing or explosion. Among these issues, one of the most well-known and persistent problems is mode collapse — where the generator fails to capture the diversity of the dataset and only produces limited types of outputs.

2.2 Problem of Mode Collapse and Diversity in GANs

Mode collapse occurs when the generator learns to produce only a few types of outputs, regardless of the input noise vector. This means that although the samples may look realistic, they don't represent the full range of variations found in the training data. For example, if a GAN is trained on a dataset with ten different classes, it might only learn to generate images from three or four of them. This significantly limits its usefulness for tasks that require complete data coverage.

There are two related aspects of this issue: (i) Low coverage — the generator completely misses certain regions or "modes" in the data distribution, such as minority classes or rare features. (ii) Low diversity — even when the generator does capture some modes, the samples may be overly similar to one another, reducing variation within those classes. Both problems reduce the practical utility of GANs in real-world settings, especially in fields where completeness and fairness of data representation matter.

Several causes of mode collapse have been identified. A common one is when the discriminator becomes too strong too quickly during training, making it hard for the generator to learn anything useful — resulting in vanishing gradients. The optimization landscape of GANs is also highly non-convex and unstable, making convergence difficult. Lastly, the GAN objective itself doesn't explicitly enforce that all modes of the data distribution must be learned. So unless the generator is somehow encouraged to explore and capture rare modes, it tends to focus on producing only what it can generate reliably.

Over the years, many solutions have been proposed to mitigate this problem — such as architectural changes, modified loss functions, and alternative sampling strategies. Yet, mode collapse remains an open challenge in GAN research. The Self-Diagnosing GAN (Dia-GAN) takes a different route by integrating underrepresented sample detection directly into the training loop. Instead of relying solely on fixed sampling or loss-based tricks, it actively tracks which real samples are being ignored by the generator and adjusts their importance through weighted sampling. This approach aims to improve the generator's ability to cover the data manifold more thoroughly, and in turn, generate more diverse and representative outputs.

2.3 Mathematical Basis of Self-Diagnosing GANs

The Self-Diagnosing GAN (Dia-GAN) introduces a novel training paradigm that leverages internal statistics of the discriminator to improve the learning process. The central idea is to diagnose which real samples are being poorly modeled by the generator and to then emphasize those samples during training. This is achieved by computing statistics over the discriminator's outputs, which indirectly reflect how well the generator covers different regions of the data distribution.

The assumption is that samples that are consistently underrepresented — i.e., those for which the generator fails to produce convincing counterparts — will cause the discriminator to confidently classify them as real. By tracking this behavior over time, Dia-GAN uses the discriminator not just as a training signal, but also as a diagnostic tool.

2.3.1 Log-Density Ratio [Yanik]

The foundation of the approach is based on the concept of the log-density ratio. The discriminator $D(x)$ is trained to output the probability that a given input x is a real sample from the true data distribution. By Bayes' theorem, this probability can be expressed as:

$$D(x) = \frac{p(x \mid \text{real}) \cdot p(\text{real})}{p(x \mid \text{real}) \cdot p(\text{real}) + p(x \mid \text{fake}) \cdot p(\text{fake})} \tag{2}$$

In most GAN setups, real and fake samples are shown to the discriminator in equal proportion, so we assume $p(\text{real}) = p(\text{fake}) = 0.5$. Substituting this into the equation simplifies it to:

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)} \tag{3}$$

This is the form of the optimal discriminator under the original GAN formulation. Using this, we can derive the log-density ratio, which tells us how much more likely a sample is to come from the real data distribution than from the generated one:

$$\text{LDR}(x) = \log\left(\frac{p_{data}(x)}{p_{gen}(x)}\right) \tag{4}$$

Substituting the optimal discriminator into this expression, we get:

$$\text{LDR}(x) = \log\left(\frac{D(x)}{1 - D(x)}\right) \tag{5}$$

This formulation is very useful in practice because it allows us to estimate the log-density ratio directly from the discriminator output without needing to compute or know $p_{data}(x)$ or $p_{gen}(x)$ explicitly.

A positive LDR indicates that $p_{data}(x) > p_{gen}(x)$, meaning the generator is underrepresenting this sample — it is more likely to come from the real distribution than from the generator. Conversely, a negative LDR implies the generator is overproducing similar samples, potentially causing redundancy or overfitting.

By computing and tracking these LDR values over the course of training, Dia-GAN is able to dynamically assess which samples are being ignored or poorly modeled and use this information to adjust sampling probabilities accordingly.

2.3.2 LDRM and LDRV [Yanik]

Since single-step LDR values can be noisy or unstable across training iterations, Dia-GAN proposes aggregating them over time to produce more robust statistics. This is done using the Log-Density Ratio Mean (LDRM) and Log-Density Ratio Variance (LDRV), which are computed for each real sample across multiple past epochs. These metrics capture how consistently a sample has been underrepresented by the generator and how uncertain this underrepresentation is over time.

The Log-Density Ratio Mean (LDRM) is calculated as:

$$\mathrm{LDRM}(x) = \frac{1}{|T|} \sum_{k \in T} \mathrm{LDR}(x)_k \tag{6}$$

where T represents a window of previous training steps, and $\mathrm{LDR}(x)_k$ is the LDR value of sample x at step k. A higher LDRM indicates that the sample has consistently appeared more frequently in the real data than in the generated data, suggesting persistent underrepresentation. In the context of mode collapse, high LDRM values help identify which modes are not being covered properly by the generator.

The Log-Density Ratio Variance (LDRV) captures how much the LDR values of a sample fluctuate over time:

$$\mathrm{LDRV}(x) = \frac{1}{|T|-1} \sum_{k \in T} \left(\mathrm{LDR}(x)_k - \mathrm{LDRM}(x)\right)^2 \tag{7}$$

This measures the stability of the generator's behavior toward a particular sample. If LDRV is high, the model is inconsistent in how it handles the sample — sometimes representing it well, and other times failing. If LDRV is low, the model consistently treats the sample the same way (whether that's good or bad).

2.3.3 Score-Based Weighted Sampling [Yanik]

Dia-GAN uses the LDRM and LDRV statistics to assign an importance score to each sample. This score determines how likely a sample is to be included in the training batches. The sampling score $s(x)$ is defined as:

$$s(x) = \mathrm{LDRM}(x) + k \cdot \sqrt{\mathrm{LDRV}(x)} \tag{8}$$

Here, k is a hyperparameter that controls how much the variance contributes to the score. This formulation ensures that both consistently underrepresented and unstable samples are emphasized. Samples with high LDRM (meaning they are regularly ignored) or high LDRV (meaning the model is uncertain about them) are assigned higher sampling probabilities.

During training, instead of uniformly sampling from the training dataset, Dia-GAN constructs mini-batches using a weighted sampling strategy based on these scores. This adaptive sampling mechanism increases the chance that hard-to-learn, rare, or inconsistent samples appear in the generator's training loop, thereby improving mode coverage and diversity.

2.4 Description of Dataset Used [Gecim]

In the original Dia-GAN paper, the authors evaluate their method on several datasets with varying complexity and diversity — including CIFAR-10, CelebA, and Colored MNIST. Each dataset presents different challenges in terms of mode representation and diversity, making them useful for benchmarking the effectiveness of Dia-GAN.

CIFAR-10 is a well-known image classification dataset consisting of 60,000 color images (32x32 pixels) divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It provides a reasonably balanced and moderately complex distribution, making it a good testbed for evaluating generative models in low-resolution scenarios. In our project, we aim to finalize experiments using this dataset.

CelebA is a much larger dataset containing over 200,000 images of celebrity faces, each annotated with 40 binary attributes (such as "smiling", "wearing glasses", "bald", etc.). Due to the dataset's size and high resolution, it provides a more challenging environment for training GANs and detecting underrepresented visual traits. However, given our limited computational resources, we were not able to include CelebA in our experiments.

In addition to these, we used several toy datasets to isolate and demonstrate specific behaviors of Dia-GAN. For example, we created a synthetic 2D Gaussian Mixture dataset with known mode structure, which allowed us to visualize and validate how well LDR-based metrics highlight collapsed or ignored modes. These simpler datasets were especially useful for debugging our implementation and illustrating the effectiveness of the Dia-GAN framework in a more controlled setting.

2.5 Review of Related Work [Yanik]

Several techniques have been proposed in recent years to address the problem of mode collapse and improve sample diversity in GANs. These methods approach the issue from different angles, including post-processing, loss weighting, input manipulation, and architectural changes. In this section, we briefly review some of the most relevant approaches and highlight how Dia-GAN differs from or complements them.

2.5.1 Discriminator Rejection Sampling (DRS)

[AOD$^+$18] introduced Discriminator Rejection Sampling (DRS), a post-processing technique that uses the discriminator's output to filter out low-quality or low-confidence generated samples. The main idea is to sample from the generator as usual and then retain only those outputs that are assigned a high probability of being real by the discriminator. This helps improve sample fidelity and can boost performance on downstream tasks. However, DRS is purely a filtering mechanism applied after training — it does not influence the learning process of the generator or help it recover missing data modes. In contrast, Dia-GAN incorporates diagnostic feedback into training itself, allowing the model to adapt as it learns.

2.5.2 GOLD: Reweighting Fake Samples

The GOLD method [MZZ$^+$19] takes a different route by estimating the LDR for fake samples and using it to reweigh the generator's loss. This means that fake samples which are far from the real distribution (i.e., high LDR) are emphasized during training, pushing the generator to focus on improving those areas. While this technique improves the quality of generated samples within the existing data modes, it does not directly address mode coverage — especially if the generator never samples certain regions in the first place. Dia-GAN takes the opposite approach by reweighting real samples rather than fake ones, thereby encouraging the generator to learn from parts of the real distribution it tends to ignore.

2.5.3 PacGAN

PacGAN, introduced by [LKFO18], is another technique designed to combat mode collapse by modifying the discriminator. Instead of feeding the discriminator single samples, PacGAN presents it with groups (or "packs") of samples, which enables it to detect lack of diversity within the batch. This indirectly pressures the generator to produce more diverse outputs. While PacGAN is effective, it requires architectural changes to the discriminator and increases memory usage due to larger input batches. In contrast, Dia-GAN makes no architectural modifications and instead focuses on improving sample selection during training.

2.5.4 Inclusive GAN

[YZZ$^+$20] proposed Inclusive GANs, which integrate ideas from both Variational Autoencoders (VAEs) and GANs. Their method uses a hybrid objective that includes reconstruction loss (like in VAEs) alongside the adversarial loss, helping the generator cover more of the data space. While this improves mode coverage, it adds extra complexity to the training procedure and requires balancing multiple loss terms. Dia-GAN avoids this by working entirely within the standard GAN framework, relying only on discriminator statistics and adaptive sampling.

2.5.5 Top-k Training

[ZXL$^+$19]Top-k training is another strategy that modifies how generator updates are computed. In this approach, only the top-k generated samples (i.e., those that the discriminator scores highest) are used for backpropagation. This can improve stability and precision by focusing on the most realistic samples. However, it may hurt diversity by discarding other potentially useful or diverse samples. Dia-GAN, on the other hand, emphasizes *real* samples through adaptive weighting rather than discarding generated ones, which helps preserve diversity while still guiding the model toward underrepresented areas of the data.

In addition to the Dia-GAN training strategy, we also aim to implement Discriminator Rejection Sampling (DRS) as a post-processing step in our experiments. This will allow us to compare a training-time intervention (Dia-GAN) with a post-training filtering method (DRS), and potentially combine both approaches to create a more effective model that benefits from improved sample diversity and higher output fidelity.

3 Methods

3.1 Specializing Dia-GAN to Our Setting [Gecim]

In this project, we implemented the Self-Diagnosing GAN (Dia-GAN) framework by adapting it to a simplified but effective setup tailored to our resource limitations. Our goal was to preserve the core theoretical contributions of the original paper—namely, the use of Log-Density-Ratio Mean (LDRM) and Log-Density-Ratio Variance (LDRV) to identify and emphasize underrepresented real samples—while ensuring that the architecture and training loop remained lightweight, reproducible, and easy to experiment with under constrained environments like Google Colab.

3.1.1 Architecture

Our implementation is based on a simplified version of Spectral Normalization GAN (SNGAN) [MKKY18], a GAN variant known for its training stability and improved sample quality. We use spectral normalization on all convolutional layers in the discriminator to ensure Lipschitz continuity, which helps with stable gradient flow during training. However, given our limited resources, we reduced the number of layers and overall model size to speed up experimentation and lower memory usage. We also adapted the architecture to work with low-resolution 28×28 images (e.g., MNIST and Colored MNIST), instead of higher-resolution datasets that SNGAN was originally designed for.

The generator takes as input a 100-dimensional noise vector sampled from a standard normal distribution. This vector is projected into a dense layer, reshaped into a feature map, and passed through a series of ConvTranspose2d layers for upsampling. Each block includes BatchNorm and ReLU activations, except for the final layer, which uses a `tanh` activation to produce output images in the range [-1, 1], matching our normalized dataset inputs.

The discriminator consists of three convolutional blocks with increasing channel depth (64, 128, 256). Each block is followed by LeakyReLU activations. Spectral normalization is applied to every convolutional and linear layer. We do not apply a sigmoid activation at the end of the discriminator; instead, we use PyTorch's `BCEWithLogitsLoss`, which combines the sigmoid operation and binary cross-entropy loss into a single, more numerically stable function.

This simple yet stable architecture allowed us to conduct a wide range of experiments while maintaining fast iteration speed, which was critical given our time and hardware constraints.

3.2 Training Procedure [Gecim]

We follow the three-phase training strategy used in Dia-GAN, which separates the training process into a warm-up phase, a diagnosis phase using score-based sampling, and a final post-processing step with Discriminator Rejection Sampling (DRS).

3.2.1 Phase 1: Baseline Training

In the first phase, which spans approximately 90% of the total training epochs by default but could vary depending on the dataset. The model is trained using standard uniform sampling. That is, each real sample from the dataset is equally likely to be selected, without any weighting or prioritization. This warm-up phase gives the generator time to learn a rough approximation of the data distribution and stabilize before any diagnostic sampling is introduced. During this stage, we also begin tracking the Log-Density Ratio (LDR) values for each sample, although they are not yet used to influence the training batches. This ensures that we have a history of LDRs available by the time score-based sampling starts.

3.2.2 Phase 2: Score-Based Weighted Sampling

Once the baseline training phase is complete, we transition to score-based weighted sampling using the LDRM and LDRV metrics. At this point, the generator has already learned to reproduce the dominant modes in the data. The goal of this phase is to help it focus on harder-to-learn and underrepresented samples.

At the end of each epoch, we compute a sampling score $s(x)$ for each real sample x using the following formula:

$$s(x) = \text{LDRM}(x) + k \cdot \sqrt{\text{LDRV}(x)} \tag{9}$$

Here, k is a tunable hyperparameter that balances the contribution of the variance component. This score reflects both the persistence (via LDRM) and uncertainty (via LDRV) of a sample being underrepresented.

To prevent the sampling probabilities from becoming too skewed, we apply clipping using `min_clip` and `max_ratio` thresholds. After clipping, the scores are normalized to form a valid probability distribution. This distribution is then used to create a `WeightedRandomSampler` for PyTorch's `DataLoader`, allowing us to sample mini-batches with preference toward high-score samples. This dynamic sampling mechanism ensures that rare or unstable examples are seen more frequently, which encourages the generator to model them better and improves mode coverage.

3.2.3 Phase 3: Post-training Evaluation

After training is complete, we apply Discriminator Rejection Sampling (DRS) as a post-processing step to further improve the fidelity and diversity of generated samples. The idea behind DRS is to use a trained auxiliary discriminator to re-score generated images. For each generated sample, the auxiliary discriminator estimates how confidently it believes the image is real. These scores are then converted into acceptance probabilities.

Only samples with high enough discriminator scores are kept, while others are discarded. This process does not modify the generator itself but acts as a filter to improve the overall quality of the output. While DRS is independent of the Dia-GAN training strategy, combining it with score-based training offers a way to balance both training-time diversity (via LDR-based weighting) and inference-time quality (via rejection sampling).

3.3 Model Variants and Experimental Comparison [Gecim]

To evaluate the effectiveness of the Dia-GAN framework, we implemented and compared several model variants. Each variant uses the same base architecture and training settings to ensure a fair comparison, and the only differences lie in the sampling strategies and post-processing steps. This setup helps us isolate the impact of LDR-based diagnostics and rejection sampling.

3.3.1 Baseline GAN

The baseline model serves as our control. It uses the same generator and discriminator architecture as the Dia-GAN version, but with no score-based sampling or post-processing. All real data samples are selected using uniform random sampling throughout the entire training process. This variant allows us to measure how much gain (in terms of diversity, coverage, or sample quality) comes purely from the additional components introduced in Dia-GAN.

3.3.2 Dia-GAN

This variant incorporates all the core ideas of the Self-Diagnosing GAN framework. It includes:

- Score-based sampling activated after the warm-up phase (Phase 1)
- Log-Density Ratio Mean (LDRM) and Variance (LDRV) tracking for each real sample
- Dynamic computation of sampling probabilities based on the history of LDR values (we use a sliding window of 50 past steps)

By comparing this model to the baseline, we directly test the central claim of the Dia-GAN paper: that prioritizing underrepresented and uncertain samples during training leads to better coverage of the data distribution and improves diversity.

We also plan to run ablation-style comparisons by varying the hyperparameter k in the sampling score formula. This helps us understand how sensitive Dia-GAN is to the balance between mean and variance in the sampling decisions.

3.3.3 Dia-GAN and Baseline GAN with DRS

In addition to training-time changes, we also evaluate the effect of Discriminator Rejection Sampling (DRS) as a post-processing step. We apply DRS to both the Baseline GAN and Dia-GAN variants to assess whether filtering generated samples based on a separate discriminator's confidence can improve sample fidelity and reduce noise.

This comparison serves two purposes:

- It lets us evaluate how much DRS alone can improve the outputs of a regular GAN.
- It allows us to see whether DRS adds further benefit on top of Dia-GAN, or if Dia-GAN already captures most of the gains during training.

Together, these experiments give us a clearer picture of the individual and combined contributions of score-based sampling and post-hoc filtering. The goal is to understand not just whether Dia-GAN improves performance, but how it compares to more established alternatives when operating under the same resource constraints.

3.4 Evaluation Metrics and Success Criteria [Yanik]

To define the success of Dia-GAN compared to the baseline, we adopt a quantitative evaluation protocol using three complementary metrics: Frechet Inception Distance (FID), precision, and recall. These metrics allow us to assess both the visual quality and the diversity of the generated samples.

Frechet Inception Distance (FID) measures the similarity between the distributions of real and generated image features. It compares the mean and covariance of features extracted from a pretrained network. A lower FID score indicates that the generated images are closer to real ones, both in quality and diversity.

Precision and recall are used to analyze the generator's coverage of the data distribution more directly. For this purpose, we use a classifier-based evaluation instead of relying on the Inception network, which is more suitable for high-resolution natural images. In our case, we train a custom classifier on our datasets and use its feature embeddings for metric computation.

Precision measures the proportion of generated samples that lie close to real data in feature space, indicating visual realism. Recall measures how many real samples are "covered" by the generator — in other words, how much of the real data manifold the model can reproduce. Both metrics ideally approach 1, although in practice there's usually a trade-off between them.

To compute these metrics, we follow this procedure:

- Generate synthetic samples from the trained GAN.
- Extract features from both real and generated datasets using a pretrained classifier.
- Compute FID by calculating the Fréchet distance between the distributions of these features.
- Compute precision and recall using k-nearest neighbors in the normalized feature space. We define sample membership based on a fixed distance threshold ε. For precision, we check how many generated samples are close to real samples. For recall, we do the reverse.

In our implementation, we use $\varepsilon = 0.4$ and $k = 3$ to define neighborhood proximity for both metrics. This framework allows us to quantitatively assess whether improvements are due to better coverage (higher recall), improved quality (higher precision), or both.

A successful Dia-GAN model is expected to achieve higher recall, similar or higher precision, and a lower FID compared to the baseline. This would demonstrate its ability to capture underrepresented data modes while maintaining or improving sample quality.

Compared to the original Dia-GAN paper, which uses a StyleGAN2 backbone and evaluates on high-resolution datasets such as FFHQ and AFHQ, our implementation is deliberately scaled down. We work with lower-resolution datasets like CIFAR-10 and Colored MNIST, and use a lighter SNGAN-style architecture. Additionally, our precision and recall metrics are based on custom classifier embeddings rather than Inception-V3. These changes make our experiments more computationally feasible while still capturing the key motivations of the Dia-GAN framework — namely, improving diversity through better mode coverage.

3.5 Project Planning, Resource Allocation, and Team Organization [Yanik]

At the start of the project, we divided the three-week time frame into three main phases: (1) understanding and analyzing the Dia-GAN method along with related literature, (2) implementing the method and running diagnostic experiments, and (3) evaluating the results and compiling the report. Our original plan was to reproduce all comparison baselines from the Dia-GAN paper; however, due to time and resource constraints, we decided to focus on the core ideas and scale down accordingly.

We worked as a two-person team and split responsibilities based on individual interests and strengths. One of us focused more on implementing the training pipeline and the model architectures, while the other handled the metric tracking (LDRM, LDRV), experimental design, and evaluation. We used GitHub for version control, Google Docs and Overleaf for collaborative documentation, and had regular video calls to sync up, discuss progress, and solve issues as they came up.

For computational resources, we relied on Google Colab notebooks with GPU acceleration. Since Colab has limits on session duration and memory, we reduced dataset sizes and simplified the architecture to keep training feasible. While the original paper uses StyleGAN2 and high-resolution datasets like CelebA and FFHQ, we focused on CIFAR-10, Colored MNIST, and a custom 2D Gaussian mixture dataset — which allowed us to explore the key ideas without hitting resource limits.

3.6 Challenges and Their Resolution [Yanik]

We encountered several challenges throughout the project, both on the implementation side and due to infrastructure limitations. One of the more difficult issues was tracking LDRM and LDRV values across training without running into memory problems. Since these metrics require storing discriminator outputs per sample over time, we implemented a rolling buffer with a fixed history window (50 steps). This helped reduce memory usage while still giving a decent approximation of sample behavior.

Another challenge came when we switched from uniform sampling to score-based weighted sampling. Doing this abruptly led to instability — gradients started to oscillate and the generator struggled to converge. To fix this, we introduced the sampling weights more gradually and clipped scores to avoid extreme values. This stabilized the training process and made sampling more consistent.

On the infrastructure side, long experiments on Colab were often interrupted due to timeouts or memory errors. To deal with this, we kept training runs short, saved intermediate results often, and ran lightweight experiments in parallel to test hyperparameters and variants more efficiently.

We also intended to evaluate our models on the CIFAR-10 dataset but were ultimately unable to do so due to runtime and memory constraints on Google Colab. The higher resolution and increased complexity of CIFAR images, combined with longer convergence times and more GPU-intensive training, made full experimentation on CIFAR impractical within our compute budget. Additionally, the persistent session interruptions and lack of persistent disk in Colab discouraged the repeated long runs necessary to reach convergence on CIFAR.

Our Discriminator Rejection Sampling (DRS) results also did not show consistent improvements. We hypothesize this could be due to several factors: (1) the auxiliary discriminator may have underfit or overfit the real/fake decision boundary given the limited training epochs and noisy samples; (2) our evaluation datasets may have lacked sufficient diversity, leading to ambiguous distinctions during rejection; and (3) low-resolution image settings (e.g. MNIST) may not benefit as much from post-hoc filtering as higher-resolution settings where visual realism plays a larger role. These issues highlight areas for further exploration and suggest that DRS performance is highly sensitive to architectural and dataset conditions.

Finally, we weren't able to replicate every baseline from the original paper due to time and GPU limits. Instead, we designed smaller-scale but focused comparisons: baseline GAN vs. Dia-GAN, and both with/without DRS. These setups still reflect the core motivations of the paper and gave us meaningful insights — all while staying within realistic, reproducible conditions.

Finally, we weren't able to replicate every baseline from the original paper due to time and GPU limits. Instead, we designed smaller-scale but focused comparisons: baseline GAN vs. Dia-GAN,

and both with/without DRS. These setups still reflect the core motivations of the paper and gave us meaningful insights — all while staying within realistic, reproducible conditions.

4 Experiments and Results

4.1 Evaluating the Effectiveness of LDRM on a Gaussian Mixture Dataset [Gecim]

To investigate the dignostic capabilites of Log Density Ratio Mean (LDRM), we begin by replicating the toy experiment. The data we have used consists of 25 modes of Gaussian Mixture. We arrange these in a 5×5 grid spanning from -2 to 2 in both directions. Each Gaussian component contains 1000 samples drawn from a normal distribution with a standard deviation of 0.05. This results in a clearly structured, multimodal dataset that serves as a benchmark to study mode collapse.

We have implemented very basic MLP based Generator and Discriminator, which fits well with our 2 dimensional toy set-up. We train the GAN for 300 epochs using the Adam optimizer with learning rate, and standard GAN loss (binary cross-entropy). During training, we compute the Log Density Ratio (LDR) for real samples from each Gaussian mode.

To capture temporal dynamics and smooth fluctuations, we compute the LDRM for each mode using a window size of 50 epochs. This is aggregated into a heatmap with class IDs (mode indices) on the y-axis and training epochs on the x-axis. The intensity in the heatmap reflects how confidently the discriminator classifies real samples from each mode. To supplement the LDRM heatmap, we also generate and visualize samples from the generator every 50 epochs. This helps qualitatively assess how well the generator covers all modes, and allows cross-checking with the LDRM trends.

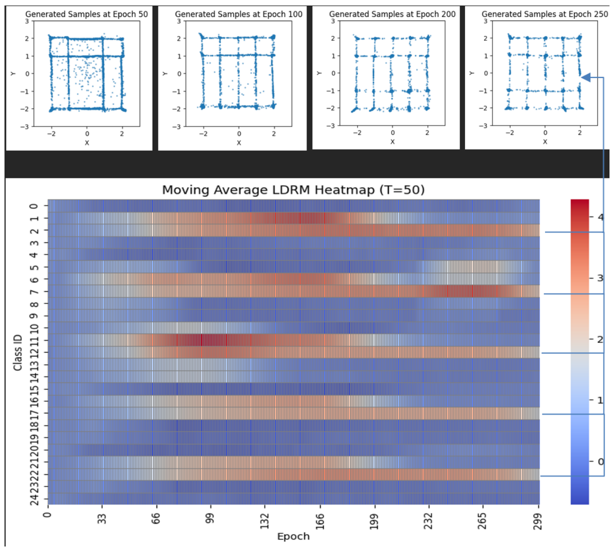


Figure 1: Heatmap of LDRM and generated samples throughout training.

In the figure 1, we have visualized the aforementioned plots on top of each other. The labels of the modes start from the lower left corner (0) than continues column by column till the upper right corner (24). As one can see from the generated images GAN struggles to learn the middle row, corresponding to the modes 2, 7, 12, 17 and 22, throughout the training and this has been effectively captured by our LDRM heatmap. When we look at the beginning to middle stages of training the second row from the bottom, also had difficulties by the discriminator. But after the training step 200 those modes are recovered and the heatmap for those labels turn back to blue. This indicates that LDRM of the data instances can be used to detect the regions of data manifold not yet covered by the model, even without looking at the generated samples.

4.2 LDRV as a Diagnostic for Minority Group Detection [Gecim]

In addition to detecting modes in multi-model data, we further evaluate the Log-Density Ratio Variance as a tool for identifying minority groups. The experiment in this subsection aims to validate the hypothesis from the Self-Diagnosing GANs paper that minority groups tend to exhibit greater variability in their LDR signal across training steps.

As toy data, we have constructed a synthetic 2D dataset by sampling 10000 pints from an isotropic Gaussian with standard deviation $\sigma = 3.0$. We then assign binary class labels based on their distance of each point from the origin. This yields a natural imbalance, where the minority class resides in the tails of the distribution and comprises a much smaller portion of the data.

We train the same basic GAN architecture as in the previous subsection. During training, we monitor the LDR values for real samples from both major and minor groups at every epoch. These values are stored separately per class and then used to compute the LDR Moving Average (LDRM) as well as the LDRV, defined as the variance of LDR over time for each group. Epoch-wise sample generation also would help visualize how well the generator learns to replicate both the core and tails of the distribution. At the end of the training, we compute the variance of LDR values (LDRV) across epochs for both groups. This quantifies how unstable or inconsistent the discriminator's confidence is across training. As reported in Table 1 of the original paper, the minority group is expected to show a higher LDRV,

Group	Gaussian ($\sigma = 3.0$)
Major	0.019
Minor	0.185

Table 1: Averaged LDRV of major minor groups on Gaussian dataset

This supports the claim that minority groups are less consistently learned by the generator and discriminator pair, and that LDRV provides a sensitive signal for detecting such failures. A higher LDRV for the minority group suggests that the GAN exhibits unstable behavior toward that group over the course of training, potentially due to poor sample coverage or insufficient representation.

4.3 Dia-GAN Hyperparameter Analysis [Gecim]

In this section we would like to investigate the optimal hyperparameters of the self diagnosing GAN. We will again be using the (5×5) Gaussian mixture dataset for the experiments. First we will be focusing on the phase 1 ratio, where we collect the LDR values and its modes for the underrepresented sample emphasis on the second phase of the training. This ratio basically tells the model how much of the training should be done as in a standard GAN training. Than the hyperparameter k.

4.3.1 Hyperparameter Phase 1 Ratio: Fraction of training step spent in Phase 1

In this experiment, we investigate the impact of the phase 1 ratio hyperparameter on the performance of Dia-GAN. This parameter controls the fraction of training time spent in Phase 1, where the GAN is trained using standard adversarial loss without any diagnostic regularization. The remaining time (1 - phase 1 ratio) is allocated to Phase 2, where Dia-GAN introduces the LDR-based diagnostic loss to penalize underrepresented regions.

The goal of this experiment is to understand how varying the length of the diagnostic-free phase effects the final performance of the model on the performance metrics Precision Recall and Mode Coverage. We have implemented the mode coverage, for a better intuition, which counts the generated samples in the vicinity of the given centers and if a given threshold is reached, indicates that the mode is covered. A longer Phase 1 might help stabilize the GAN before diagnostics are introduced. A shorter Phase 1 on the other hand gives more time for the generator to benefit from the score based weighted sampling. We would like to find the sweet spot that balances these benefits. We vary the Phase 1 Ratio in the set 0.3,0.5,0.8,0.9 while keeping the diagnostic loss weight fixed at k=0.5, based on the previous experiment. Each configuration is run 5 times to average out randomness in training.

End result of this sub task is given in the 2 and visualized in the 2. As expected mode coverage and recall how the same behavior as they are describing the same thing form different point of views

Phase 1 Ratio	Precision	Recall	Mode Coverage
0.3	0.52748	0.718160	12.0
0.5	0.63918	0.838336	19.4
0.8	0.57300	0.802144	19.4
0.9	0.45274	0.795472	18.6

Table 2: Effect of Phase 1 Ratio on Model Performance

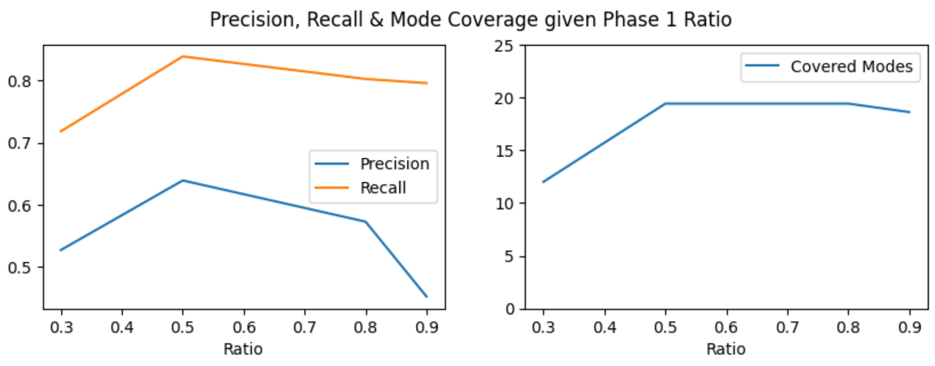


Figure 2: Phase 1 Ratio Performance

4.3.2 Hyperparameter k: Emphasis on LDRV

To further analyze the behavior and effectiveness of the DiaGAN training strategy, we conduct a hyperparameter sensitivity study focusing on the role of the k parameter. This experiment aims to empirically investigate how varying the value of $k \in 0.0, 0.3, 0.5, 1.0, 2.0$ affects Precision, Recall and Mode Coverage. We have tested the hyperparameter with a fixed Phase 1 Ratio (0.5) which we decided on in the previous section.

For $k = 0$ we are basically eliminating LDRM from influencing the training. But the algorithm still benefits from the phase separation as the LDRM still helps reduce the mode collapse which we have shown in the sections before. For low k-values 0.3 to 0.5 we expect o see an increase in both recall and mode coverage as LDRV starts to emphasize the underrepresented samples and the data manifold gets a better coverage. In these values we do not expect to see decrease in precision yet. But for higher values of k, the LDRV dominates and forces the model to explore minorities. This might result in a sacrifice of precision for better coverage trade-off. We would like to find a sweet spot where this trade-off results in the optimal approximation of the data manifold.

k	Precision	Recall	Mode Coverage
0.0	0.63760	0.819712	15.6
0.3	0.60038	0.818848	16.6
0.5	0.62334	0.788272	17.0
1.0	0.54594	0.846656	20.6
2.0	0.52100	0.804464	19.4

Table 3: Caption

With the table 3 and the plot 3. We have decided on the optimal k value of $0.3 - 1.0$. Evn though $k = 0.0$ also indicate good learning with recall and precison metrics but Mode coverage says otherwise.

4.4 Comparisons [Yanik]

In this section we will conduct our experiments to test our proposed training algorithm against traditional ones. These will cover GMM, MNIST and Colored MNIST, which allows us to experiment with

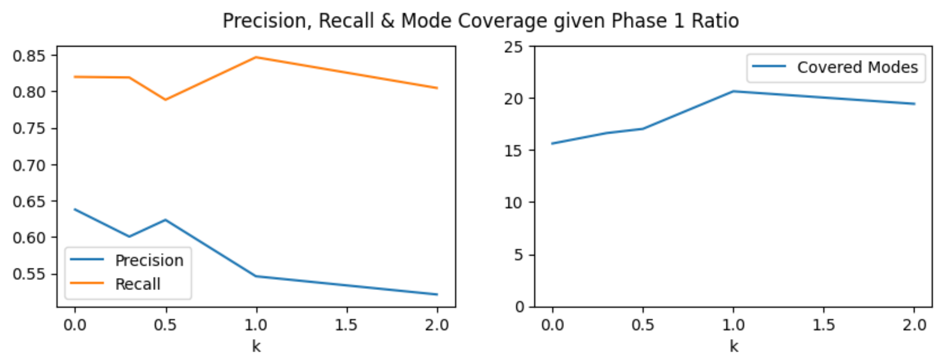


Figure 3: Effect of k in Model Performance

the minority percentage as well. As our proposition tackles the training itself, the architectures and standard hyperparameters like number of epochs learning rate are the same. These will be declared for comparison.

4.4.1 Gaussian Mixture Model

With the optimal hyperparameter we will now compare the standard GAN training algorithm with our proposed self-diagnosing training algorithm. We will first conduct our experiments on our reliable and fast Gaussian Mixture Model dataset with 5×5 modes. We have trained both of the models for 100 epochs with the learning rate of 0.0003. The generator gets a noise of dimension 2 (latent dimesnsion) to approximate the modes. In the table 4 we can see that our results support our proposition as both data coverage and overall data quality were improved.

Model	k	Precision	Recall	Mode Coverage
Basic GAN	-	x	x	x
Dia-GAN	0.3	0.64572	0.822784	17.4
Dia-GAN	0.5	0.63942	0.858448	18.2
Dia-GAN	1.0	0.49870	0.861648	21.0

Table 4: Basic GAN vs Dia-GAN on GMM

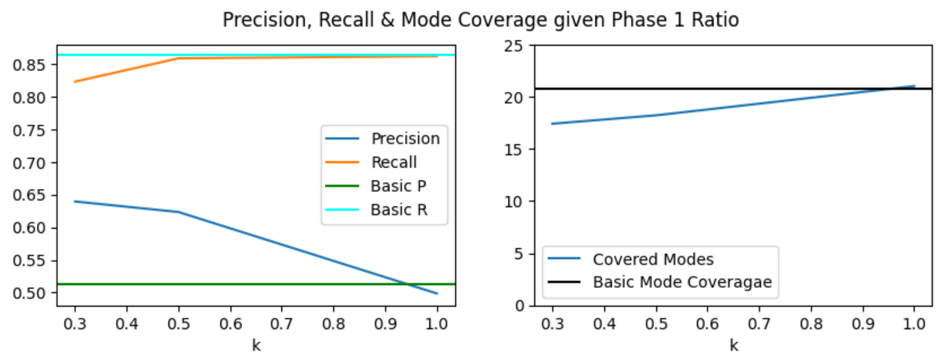


Figure 4: Basic GAN vs Dia-GAN on GMM

We have now with this step concluded that a k between $0.3 - 0.5$ is optimal since the precision declines after the threshold of around 0.5.

4.4.2 MNIST

After validating the effectiveness of our self-diagnosing training method on the 2D Gaussian Mixture dataset, we further evaluate its performance on a more complex and higher-dimensional dataset

MNIST. This real-world benchmark allows us to assess how well our approach generalizes beyond synthetic toy distributions.

In this setting, we compare the standard BasicGAN against our proposed DiaGAN, using the optimal hyperparameter values previously established from the GMM experiments — specifically, k=0.5 (for emphasis on data coverage) and Phase 1 ratio = 0.5.

To account for the more intricate structure of MNIST images, we employ deeper convolutional neural networks for both the generator and discriminator. The discriminator is a three-layer convolutional network with spectral normalization applied to all layers to promote Lipschitz continuity and stabilize training.

Both models have been trained for 50 epochs using the Adam optimizer with a learning rate of 0.0002. The generator approximates the data distribution given the latent space of dimension 100. Input images are normalized to the [-1,1] range, and the discriminator outputs raw logits (i.e., no final sigmoid), allowing flexibility in the loss design. We will form now on use an additional evaluation metric of FID with Precision and Recall.

Model	**Precision** $\uparrow$	**Recall** $\uparrow$	**FID** $\downarrow$
BasicGAN	0.374	0.371	**142.84**
DiaGAN	0.361	**0.401**	166.77

Table 5: Basic GAN and DiaGAN on MNIST (100 epochs, $k = 0.5$, phase 1 ratio=0.5)

As expected the Recall got better but Precision slightly decreased this may result from the trade-off we mentioned above.

4.4.3 Colored MNIST

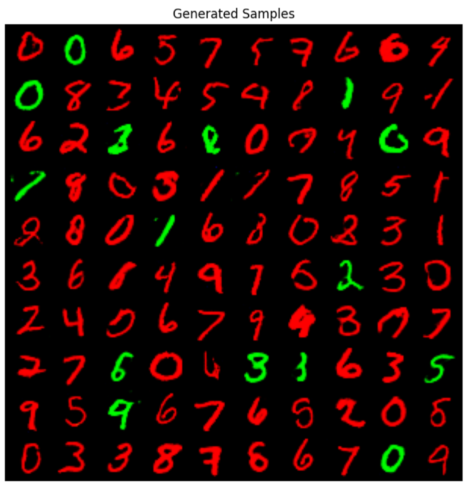
To assess how well Dia-GAN recovers underrepresented modes in a controlled imbalance setting, we conducted experiments on the Colored MNIST dataset. In this dataset, each digit class is paired with a specific color channel (red, green), but one color-digit combination is deliberately made rare (greens appearing only 20% of the time). This setup simulates a minority group in a generative modeling task, making it an ideal benchmark for testing mode coverage and bias mitigation.

Model	**FID** $\downarrow$	**Precision** $\uparrow$	**Recall** $\uparrow$
BasicGAN	606.53	0.641	0.464
Dia-GAN	608.83	0.612	0.475

Table 6: Performance comparison on Colored MNIST.

On the table 6 our results can be seen. We have also plotted 100 generated images from both of the models.

Although the quantitative metrics (FID, precision, recall) do not indicate a large performance gap between Dia-GAN and the baseline, a qualitative inspection of 100 generated samples from each model reveals a notable difference: Dia-GAN produces significantly more minority (green) digit samples than the BasicGAN. This discrepancy between visual outcome and metric scores can be attributed to several factors. First, standard recall metrics do not explicitly account for specific class-conditional mode recovery, but instead reflect global sample diversity. Second, the precision-recall evaluation relies on distances in feature space, which may not be sufficiently sensitive to subtle changes in color channels—especially in low-resolution inputs like 28×28 RGB images. Finally, the classifier embedding used for evaluation was not trained to distinguish or prioritize color information, possibly underestimating the impact of improved minority generation. This underscores the importance of combining quantitative evaluation with targeted visual analysis when studying fairness and mode coverage in generative models.



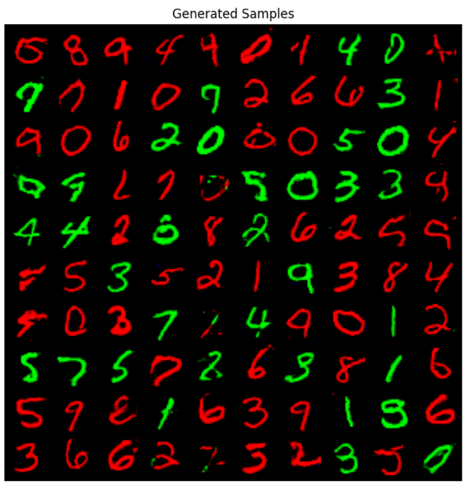


Figure 5: Left: Samples from BasicGAN. Right: Samples from Dia-GAN. Note the increased presence of green-colored digits in Dia-GAN outputs.

5 Conclusion and Outlook [Yanik]

In this project, we set out to evaluate the Self-Diagnosing GAN (Dia-GAN) framework, with a focus on its ability to detect and improve coverage of underrepresented data modes during GAN training. Our main research questions were: (1) Can the Log-Density Ratio Mean (LDRM) and Log-Density Ratio Variance (LDRV) serve as reliable diagnostics for identifying poorly learned regions of the data distribution? And (2) does score-based weighted sampling guided by these metrics actually lead to better diversity and data coverage, without heavily sacrificing sample quality?

We believe we were able to answer both questions with a reasonable degree of confidence. First, our experiments with toy datasets clearly showed that LDRM and LDRV are meaningful indicators of mode coverage and sample representation. The LDRM heatmaps successfully highlighted collapsed or missing modes in the generator's output, even without inspecting the generated images directly. Similarly, our analysis with the minority-majority setup confirmed that LDRV consistently reflected the instability in learning underrepresented samples.

On the second question, our Dia-GAN models showed measurable improvements in recall and mode coverage across multiple datasets. While this sometimes came at the cost of slightly reduced precision or higher FID scores—especially on more complex datasets like MNIST—we observed that this trade-off was often worth it in scenarios where diversity is a critical requirement.

To evaluate the performance of both BasicGAN and DiaGAN on Colored MNIST, we computed precision, recall, and Fréchet Inception Distance (FID) using features extracted from a custom MNIST-trained classifier. Unlike the conventional InceptionNet-based FID (which poorly aligns with grayscale digits), our classifier-based evaluation ensures more domain-relevant assessments. As expected, Dia-GAN achieved higher recall, indicating improved sample diversity and better coverage of the MNIST data manifold. Precision dropped slightly, consistent with the increased exploration of rare or ambiguous digit styles. This trade-off also led to a higher FID, which reflects both the broader spread of the fake feature distribution and a shift in its mean relative to real data. These results are aligned with the theoretical goals of DiaGAN: improving diversity at a manageable cost to sample fidelity, as also discussed in the original paper.

There were, of course, some limitations. Due to resource and time constraints, we had to work with smaller datasets and lighter models compared to those used in the original paper. We also did not explore every single baseline and comparison strategy from the original work. Still, we believe our scaled-down implementation captures the core idea of the method and shows that even simple models can benefit from this diagnostic-guided training.

Going forward, there are a few clear directions worth exploring. One would be to apply Dia-GAN to more complex or imbalanced real-world datasets, where under-representation is more common. Another would be to combine LDR-based sampling with architectural tricks like PacGAN or training strategies like top-k sampling to investigate whether the benefits can stack. We also noticed that the performance was quite sensitive to hyperparameters like k and phase-1 ratio, so more work could be done on dynamic tuning or adaptive schedules for these values.

To sum up, our takeaway is that diagnostic signals from the discriminator—when tracked and used carefully—can be powerful tools in improving GAN training, especially for recovering minor modes and enhancing data coverage. The Dia-GAN framework provides a promising step in this direction, and we hope our results serve as a small but useful contribution to that ongoing research.

References

[AOD$^+$18] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. October 2018.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[LKFO18] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in neural information processing systems*, pages 1498–1507, 2018.

[LKHC21] Jinhee Lee, Haeri Kim, Youngkyu Hong, and Hye Won Chung. Self-diagnosing GAN: Diagnosing underrepresented samples in generative adversarial networks. February 2021.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[MZZ$^+$19] Shuang Mo, Hang Zhao, Zhiting Zhu, Honglak Lee, Li Fei-Fei, Trevor Darrell, and Fisher Yu. An effective training framework for conditional image generation with gans. In *International Conference on Learning Representations*, 2019.

[YZZ$^+$20] Ning Yu, Shanghang Zhao, Zijian Zhang, Ashwin Acharya, David Cox, and Rogerio Feris. Inclusive gan: Improving data and distribution coverage via conditional generation. In *European Conference on Computer Vision*, pages 683–701. Springer, 2020.

[ZXL$^+$19] Han Zhang, Zizhao Xu, Honglak Li, Shaoqing Zhang, Xiaogang Wang, Dimitris N Metaxas, and Tong Zhang. Top-k training of gans: Improving the diversity of generated samples. In *Advances in Neural Information Processing Systems*, volume 32, pages 9427–9437, 2019.