Dropout-GAN: Learning from a Dynamic Ensemble of Discriminators

Gonçalo Mordido
Hasso Plattner Institute
Potsdam, Germany
goncalo.mordido@hpi.de

Haojin Yang
Hasso Plattner Institute
Potsdam, Germany
haojin.yang@hpi.de

Christoph Meinel
Hasso Plattner Institute
Potsdam, Germany
christoph.meinel@hpi.de

ABSTRACT

We propose to incorporate adversarial dropout in generative multi-adversarial networks, by omitting or dropping out, the feedback of each discriminator in the framework with some probability at the end of each batch. Our approach forces the single generator not to constrain its output to satisfy a single discriminator, but, instead, to satisfy a dynamic ensemble of discriminators. We show that this leads to a more generalized generator, promoting variety in the generated samples and avoiding the common mode collapse problem commonly experienced with generative adversarial networks (GANs). We further provide evidence that the proposed framework, named Dropout-GAN, promotes sample diversity both within and across epochs, eliminating mode collapse and stabilizing training.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; *Knowledge representation and reasoning*; Computer vision representations;

KEYWORDS

Generative adversarial networks, adversarial training, dropout

1 INTRODUCTION

Generative adversarial networks [15], or GANs, are a framework that integrates adversarial training in the generative modeling process. According to its original proposal, the framework is composed of two models - one generator and one discriminator - that train together by playing a minimax game. While the generator tries to fool the discriminator by producing fake samples that look realistic, the discriminator tries to distinguish between real and fake samples better over time, making it harder to be fooled by the generator.

However, one of the main problems with GANs is mode collapse [2, 3, 8, 20, 27], where the generator is able to fool the discriminator by only producing data coming from the same data mode, *i.e.,* connected components of the data manifold. This leads to a poor generator that is only able to produce samples within a narrow scope of the data space, resulting in the generation of only similarly looking samples. Hence, at the end of training, the generator comes short regarding learning the full data distribution, and, instead, is

only able to learn a small segment of it. This is the main issue we try to tackle in this work.

In a disparate line of work, dropout was introduced by [18] and it has been proven to be a very useful and widely used technique in neural networks to prevent overfitting [5, 11, 37]. In practice, it simply consists of omitting or dropping out, the output of some randomly chosen neurons with a probability d or dropout rate. The intuition behind this process is to ensure that neurons are not entirely dependent on a specific set of other neurons to produce their outputs. Instead, with dropout, each neuron relies on the population behavior of several other neurons, promoting generalization in the network. Hence, the overall network becomes more flexible and less prone to overfitting the training data.

The main idea of this work consists of applying the same dropout principles to generative multi-adversarial networks. This is accomplished by taking advantage of multiple adversarial training, where the generator's output is dependent on the feedback given by a specific set of discriminators. By applying dropout on the feedback of each discriminator, we force the generator to not rely on a specific discriminator or discriminator ensemble to learn how to produce realistic samples. Thus, the generator guides its learning from the varied feedback given by a dynamic ensemble of discriminators that changes at every batch.

In our use case, one can then see mode collapsing as a consequence of overfitting to the feedback of a single discriminator, or even a static ensemble of discriminators. Hence, by dynamically changing the adversarial ensemble at every batch, the generator is stimulated to induce variety in its output to increase the chances of fooling the different possible discriminators that may remain in the ensemble at the end.

Our main contributions can be stated as follows:

- We propose a novel and generic framework, named Dropout-GAN (Section 3), that trains a single generator against a dynamically changing ensemble of discriminators.
- We provide useful discussions and insights regarding the benefits of multiple adversarial training in GANs, namely the increase training stability (Section 4).
- We test our method on several datasets and show it succeeds in reducing mode collapse and promoting sample diversity across consequent epochs (Sections 5 and 6), while being competitive against other state-of-the-art methods (Section 7).

2 GENERATIVE ADVERSARIAL NETWORKS

As originally proposed [15], the standard GANs framework consists of two different models: a generator (G), that tries to capture the real data distribution to generate fake samples that look realistic, and

a discriminator (D), that tries to do a better job at distinguishing real and fake samples. G maps a latent space to the data space by receiving noise as input and applying transformations to it to generate unseen samples, while D maps a given sample to a probability p of it coming from the real data distribution.

In the ideal setting, given enough iterations, G would eventually start producing samples that look so realistic that D would not be able to distinguish between real and fake samples anymore. Hence, D would assign $p = 0.5$ to all samples, reaching a full state of confusion. However, given the training instability inherent to GANs training, this equilibrium is hard to reach and it is hardly ever achieved in practice.

The two models are trained together and play a minimax game:

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

where $p_z(z)$ represents the noise distribution used to sample G's input and $G(z)$ represents its output, which can be considered as a fake sample originated from mapping the modified input noise to the data space. On the other hand, $p_r(x)$ represents the real data distribution and $D(x)$ represents the output of D, i.e., the probability p of sample x being a real sample from the training set.

In order to maximize Eq. 1, D's goal is then to maximize the probability of correctly classifying a sample as real or fake by getting better at distinguishing such cases by assigning p close to 1 to real images and p close to 0 to generated images. By contrast, to minimize Eq. 1, G tries to minimize the probability of its generated samples being considered as fake by D, through fooling D into assigning them a p value close to 1.

However, in practice $\log(1 - D(G(z)))$ might saturate due vanishing gradient problems in the beginning of training caused by D being able to easily distinguish between real and fake samples. As a workaround, the authors propose to maximize $\log(D(G(z)))$ instead, making it no longer a minimax game. Nevertheless, G still continues to exploits D's weaknesses in distinguishing real and fake samples by using D's feedback to update its parameters and slightly change its output to more likely trick D in the next iterations.

3 DROPOUT-GAN

We propose to integrate adversarial feedback dropout in generative multi-adversarial networks, forcing G to appease and learn from a dynamic ensemble of discriminators. This ultimately encourages G to produce samples from a variety of modes, since it now needs to fool the different possible discriminators that may remain in the ensemble. Variations in the ensemble are achieved by dropping out the feedback of each D with a certain probability d at the end of every batch. This means that G will only consider the loss of the remaining discriminators in the ensemble while updating its parameters at each iteration. Figure 1 illustrates the proposed framework.

Our initial modification to the value function V of the minimax game is presented in equation (2), where δ_k is a Bernoulli variable ($\delta_k \sim Bern(1 - d)$) and $\{D_k\}$ is the set of K total discriminators. The gradients calculated from the loss of a given discriminator D_k, are only used for the calculation of G's final gradient updates when

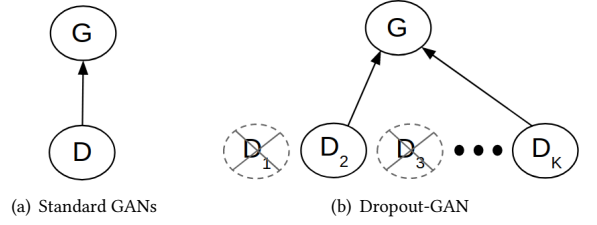(a) Standard GANs　　　　　(b) Dropout-GAN

Figure 1: We extend the original GANs framework (left) to multiple adversaries, where some discriminators are dropped out according to some probability (right), leading to only a random subset of feedback (represented by the arrows) being used by G at the end of each batch.

$\delta_k = 1$, with $P(\delta_k = 1) = 1 - d$. Otherwise, this information is discarded.

$$\min_G \max_{\{D_k\}} \sum_{i=k}^{K} V(D_k, G) = \sum_{i=k}^{K} \delta_k(\mathbb{E}_{x \sim p_r(x)}[\log D_k(x)] \\ + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_k(G(z)))]). \qquad (2)$$

There is, however, the possibility of all discriminators being dropped out from the set, leaving G without any guidance on how to further update its parameters. In this case, we randomly pick one discriminator $D_j \in \{D_k\}$ and follow the original objective function presented in equation (1), using solely the gradient updates related to D_j's loss to update G. Hence, taking into consideration this special case, our final value function, F, is set as follows:

$$F(G, \{D_k\}) = \begin{cases} \min_G \max_{\{D_k\}} \sum_{i=k}^{K} V(D_k, G), & \text{if } \exists k : \delta_k = 1 \\ \min_G \max_{D_j} V(D_j, G), & \text{if } \forall k : \delta_k = 0, \\ & \quad \text{for } j \in \{1, ..., k\} \end{cases} \qquad (3)$$

It is important to note that each discriminator trains independently, i.e., is not aware of the existence of the other discriminators, since no changes were made on their individual gradient updates. This implies that even if dropped out, each D updates its parameters at the end of every batch. The detailed algorithm of the proposed solution is presented in the Appendix.

4 IMPLEMENTATION DETAILS

In this section, we provide a detailed study of the effects of using a different number of discriminators together with different dropout rates. Moreover, we further provide insights into the consequence of splitting the batch among the different discriminators on the generator's training. The proposed framework was implemented using Tensorflow [1].

4.1 Number of Discriminators

Training instability has been noticeably reported as one of GANs biggest problems [3, 4, 8, 21, 33]. Here, we show that this problem can be eased by using multiple adversaries. This is also stated in

previous works [10, 29], however, without much detailed evidence. Furthermore, on top of increasing training stability, using multiple discriminators enables the usage of the original G loss, since there is now an increased chance that G receives positive feedback from at least one D and is able to guide its learning successfully [10].

To analyze the training procedure, we correlate the degree of training instability with the gradient updates that are being used by G to update its parameters at the end of each batch. The intuition is that if such updates are big, the parameters of the model will change drastically at each iteration. This is intuitively an alarming sign that the training is not being efficient, especially if it still occurs after several epochs of training, since G is repeatedly greatly updating its output, instead of performing slight, mild changes in a controlled fashion.

We found that when using 2, 5, and 10 discriminators, such gradients would converge to zero as training progressed, while, on the contrary, they remained high (in terms of its absolute value) when using solely one discriminator. On the other hand, we also noticed that as the number of discriminators increases, the point at which G's gradient updates start converging to 0 also increases. This suggests that using more discriminators can delay the learning process, which practically means that G will take longer to produce realistic samples. However, this is expected since G now receives more (and possibly contradictory) feedback regarding its generated samples, needing more time to utilize such information wisely.

Overall, one must then be careful not to use a sizable discriminator set that is big enough to confuse G - in Section 6, we note that using 10 discriminators can lead to this in some circumstances. On the other hand, the number of discriminators used should be big enough to generate a set in which dropping out some member's feedback shows some effect in the global framework (Sections 6 and 5 show advantages in using more than 2 discriminators in the set).

4.2 Batch Partitioning

The main purpose of splitting the batch among the different discriminators is to encourage each to specialize in different data modes. This is achieved by training them with a different subset of samples of the same size within each batch. This applies to both the fake samples produce by G and real samples retrieved from the training set. Such partitioning also allows data parallelism, diminishing the overhead caused by using more discriminators in the framework.

To further investigate the success in forcing the different discriminators to focus on different data modes, we argue that G's capacity of fooling the ensemble should decrease in such situation. This is indeed confirmed in our experiments, with G's loss being higher when the batches are split, especially later on in training where each D had enough time to focus on a single or a small subset of data modes. Thus, one can then associate the higher G loss with the generated samples now having to comply with a higher number of realistic features to be able to fool the dynamic ensemble of discriminators, with a subset of such features being used by each D to characterize a given sample as real or fake.

In practice, we increase the batch size to enable each D to be trained on the same original number of samples at each batch, similarly to when using only one D. On the other hand, G might still

have access to all samples at each batch, since it uses the feedback from the remaining discriminators to update its parameters at the end. However, having weaker discriminators by training each one of them with fewer samples than G, is not necessarily bad since they are more likely to give positive feedback to G [10, 29]. This is a result of their possible confused state that can better aid G in producing realistic looking samples than if it would continuously receive negative feedback, especially in the long run.

4.3 Dropout Rate

Dropping out the loss of a given D with a probability d before updating G's parameters is what induces variability in our framework. This forces G not to only need to fool one or even a static set of discriminators, but, instead, to fool a dynamic ensemble of adversaries that changes at every batch. Hence, performing this type of dropout can also be seen as a form of regularization, since it aims to promote more generalizability on the fake samples produced by G.

Depending on the number of discriminators used, using a small probability d of dropout might only lead to small changes in the ensemble of adversaries, making the feedback seen by G nearly constant throughout every batch. On the other hand, using a large dropout probability might lead to too much variance in the ensemble, making it difficult for G to learn properly due to the variability of the visible set.

Evidence of the correlation between the dropout rate and the quality of the generated samples is further given in Sections 5 and 6. Similarly to what was discussed in the original Dropout paper [18], we found that using $d = 0.2$ and $d = 0.5$ often led to better results, both in a qualitative and quantitative manner. Nevertheless, we also found that using any dropout rate ($0 < d \leq 1$) consistently performed better across the different datasets than when using a static ensemble of adversaries ($d = 0$).

5 EXPERIMENTAL RESULTS

We tested the effects of the different parameter settings on three different datasets: MNIST [23], CIFAR-10 [22], and CelebA [25]. We compared all possible combinations by using the different number of discriminators across the set $\{1, 2, 5, 10\}$ with each different dropout rate in $\{0.0, 0.2, 0.5, 0.8, 1.0\}$. As a baseline, mode collapse examples for the different datasets using solely 1 discriminator are illustrated in Figure 2.
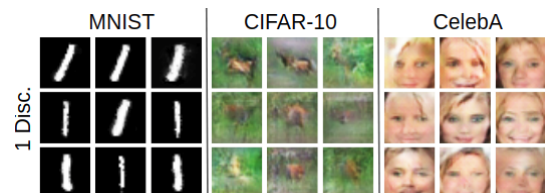

Figure 2: Examples of mode collapse occurrence while using 1 discriminator. G tends to only produce images of the number 1 on MNIST (left), deer on CIFAR-10 (middle), and blonde celebrities on CelebA (right).

All experiments were conducted using the models' architectures and training procedure described in GMAN [10], which resembles DCGAN [32]. Important to note that, even though all discriminators share the same architectures, their weights are initialized differently. Results are reported below for each dataset.

5.1 MNIST

MNIST is composed of 10 different classes of handwritten digits varying from 0 to 9. Figure 3 shows the results of Dropout-GAN on MNIST.

Figure 3: MNIST results using different combinations of the number of discriminators and dropout rates.

It is visible that the quality and variation of the produced samples increase while using dropout rate values of 0.2 and 0.5 across all different sized discriminator sets. On the other hand, the quality of the produced numbers deteriorates considerably while using high dropout rates, *i.e.,* 0.8 and 1, or no dropout rate at all. However, the quality gets slightly better when using more discriminators on such extreme end dropout rates, since G might still get enough feedback to be able to learn at the end of each batch.

5.2 CIFAR-10

To further validate our solution, we used the CIFAR-10 dataset also composed of 10 classes, consisting of different transportation vehicles and animals. Results are presented in Figure 4. Once again, we observe worst sample quality when using high or nonexistent dropout values. Moreover, there are also clear traits of mode collapsing while using no dropout rate throughout all numbers of discriminators in the set. Sharper and more diverse samples are obtained while using 0.2 or 0.5 dropout rate and a bigger number of discriminators in the set.

5.3 CelebA

We lastly tested our approach in the cropped version of CelebA, containing faces of real-world celebrities. Results are given in Figure 5. One can see that using no dropout rate leads to similar looking faces, especially when using 2 and 5 discriminators. Once more, faces produced with mid-ranged dropout values with bigger discriminator ensembles present more variety and sample quality than the rest.

Figure 4: CIFAR-10 results using different combinations of the number of discriminators and dropout rates.

Figure 5: CelebA results using different combinations of the number of discriminators and dropout rates.

6 PARAMETER EVALUATION

Since the results shown above rely heavily on subjective judgment, we now evaluate the effects of using a different number of discriminators and dropout rates on each dataset in a quantitative way. Note that the presented results are not state-of-the-art since exploring several architectural settings is not the focus of this work. Instead, by using different architectures on different datasets, the focus is solely on comparing the effect of the different parameter combinations in the end result.

6.1 Fréchet Inception Distance

We used the Fréchet Inception Distance [17] (FID) to measure the similarity between the fake images produced by G and the real images present in the full training sets. The returned distance uses the mean μ and covariance cov of a multi-variate Gaussian produced from the embeddings of the last pooling layer of the Inception-v3 model [35] for both the real data r and the generated data g. The distance is calculated as follows:

Table 1: Minimum FID obtained across 40 epochs using the different datasets. Bold scores represent the minimum FID obtained for each dataset. Underlined scores indicate the best FID within using a given number of discriminators and the different possible dropout rates regarding each dataset.

	MNIST	CIFAR-10	CelebA
1 disc.	21.71 ± 0.39	104.19 ± 0.07	53.38 ± 0.03
2 disc.; $d = 0.0$	24.88 ± 0.13	106.54 ± 0.38	52.46 ± 0.08
2 disc.; $d = 0.2$	22.34 ± 0.29	103.55 ± 0.13	46.60 ± 0.03
2 disc.; $d = 0.5$	22.08 ± 0.09	$\underline{103.20 \pm 0.05}$	$\underline{45.90 \pm 0.04}$
2 disc.; $d = 0.8$	$\underline{21.87 \pm 0.10}$	103.60 ± 0.03	46.82 ± 0.14
2 disc.; $d = 1.0$	23.56 ± 0.29	104.73 ± 0.19	51.17 ± 0.01
5 disc.; $d = 0.0$	21.47 ± 0.40	95.75 ± 0.15	45.89 ± 0.05
5 disc.; $d = 0.2$	21.70 ± 0.12	90.59 ± 0.35	$\mathbf{\underline{36.36 \pm 0.11}}$
5 disc.; $d = 0.5$	$\underline{19.25 \pm 0.12}$	$\underline{89.74 \pm 0.35}$	38.10 ± 0.54
5 disc.; $d = 0.8$	20.26 ± 0.07	90.77 ± 0.70	41.22 ± 0.24
5 disc.; $d = 1.0$	20.54 ± 0.15	95.71 ± 0.03	41.56 ± 0.18
10 disc.; $d = 0.0$	22.62 ± 0.10	99.91 ± 0.10	43.85 ± 0.30
10 disc.; $d = 0.2$	19.12 ± 0.01	91.31 ± 0.16	41.74 ± 0.14
10 disc.; $d = 0.5$	$\mathbf{\underline{18.18 \pm 0.44}}$	$\mathbf{\underline{88.60 \pm 0.08}}$	$\underline{40.67 \pm 0.56}$
10 disc.; $d = 0.8$	19.33 ± 0.18	88.76 ± 0.16	41.74 ± 0.03
10 disc.; $d = 1.0$	19.82 ± 0.06	93.66 ± 0.21	41.16 ± 0.55

$$\text{FID}(r, g) = ||\mu_r - \mu_g||_2^2 + \text{Tr}(cov(r) + cov(g)$$
$$-2(cov(r)cov(g))^{\frac{1}{2}}). \qquad (4)$$

In the original paper [17], the authors show that FID is more robust to noise and more correlated to human judgment than the previously proposed Inception Score [34]. This is naturally also applied to Mode Score [8], since this is a variant of Inception Score. Nevertheless, FID has also been shown to be sensitive to mode collapse [26], with the returned distances drastically increasing when samples from certain classes are missing from the generated set.

6.1.1 Minimum FID. Table 1 shows the best, or minimum, FID obtained by G for each dataset. Lower values indicate more similarity between the fake and real data. We ran all of our experiments for 40 epochs in total and used the same architecture described previously. To obtain the best FID across all epochs, we generated 10000 samples from G at the end of each epoch and then proceeded to calculate the FID between the set of the generated samples per epoch and the whole training set.

By analyzing Table 1, we observe that the minimum values of FID for all datasets were mostly obtained when using $d = 0.5$. However, by analyzing the local minima obtained while maintaining the same number of discriminators and only varying the dropout rate, it is also noticeable that one can also generally achieve very competitive results while using $d \in \{0.2, 0.5, 0.8\}$, depending on the number of discriminators and datasets being used. The results also show that applying dropout on multiple discriminators always leads to a better FID rather than maintaining the ensemble of discriminators static, *i.e. d = 0*, or singular, *i.e.,* using solely 1 discriminator.

6.1.2 Mean FID. We followed the same procedure and calculated the mean FID across all 40 epochs. Results are presented in Figure 6. This evaluation promotes a broader look at the stage of G at the end of every epoch, reflecting the quality and variety of the generated samples over time. The presented graphs provide a clear vision regarding the advantages of using multiple discriminators instead of solely one, with the FID being better in the first case. Using 5 or 10 discriminators with mid-range dropout rates leads to better FID results across all datasets.

The similar looking performance when using 5 and 10 discriminators can be explained by what was previously mentioned regarding G needing more time to learn from more feedback. Nevertheless, by analysis of Table 1 it is visible that, eventually, better generated samples are produced when using 10 discriminators on all datasets, even if it takes more training to reach that state. This ultimately means that by having access to more feedback, G is eventually able to produce better, varied samples in a more consistent manner over time.

6.1.3 Cumulative FID. To test the sample diversity across consecutive epochs, we calculated the FID between the set of generated samples of every consecutive epoch. This was accomplished by generating 10000 samples from G at the end of every epoch and then calculating the FID between every consequent set of generated samples. Hence, in this case, the returned FID tests the similarity between every two consecutive sets of the generated samples, evaluating the diversity of the generated samples over time. Results are shown in Figure 7.

From the analysis derived from the presented bar graphs, one can see the effect of using a different number of discriminators, with bigger sets of discriminators promoting a wider variety of generated samples across different epochs. This is generally observed across all datasets. Furthermore, it is also noticeable the benefits of using positive dropout rates to promote sample diversity.

7 METHOD EVALUATION

We now compare our approach with other existing methods in the literature. We followed the experiment with a 2D mixture of 8 Gaussian distributions (representing 8 data modes) firstly presented in UnrolledGAN [28], and further adapted by D2GAN [30] and MGAN [19]. We used the same architecture as D2GAN [30] for a fair comparison. The results are shown in Figure 8.

By the analysis of the figure, one can see that Dropout-GAN successfully covers the 8 modes from the real data, while having significantly less noisy samples compared to the majority of the other methods. Note that MGAN [19] takes advantage of a multiple generator framework, more precisely 8, plus an additional classifier network, and the architecture used is more powerful than all the other compared methods, including ours. Nevertheless, our framework manages to achieve the lowest distance and divergence measures between the real and fake data.

We further directly compare the quality of the generated samples between Dropout-GAN, GMAN [10], and original GANs [15] with the modified loss, using both 2 and 5 discriminators. Inception Score [34] was the metric used this time, with higher values correlating to better generated samples. For a fair comparison, we used the same architectures and training procedure as GMAN. Results
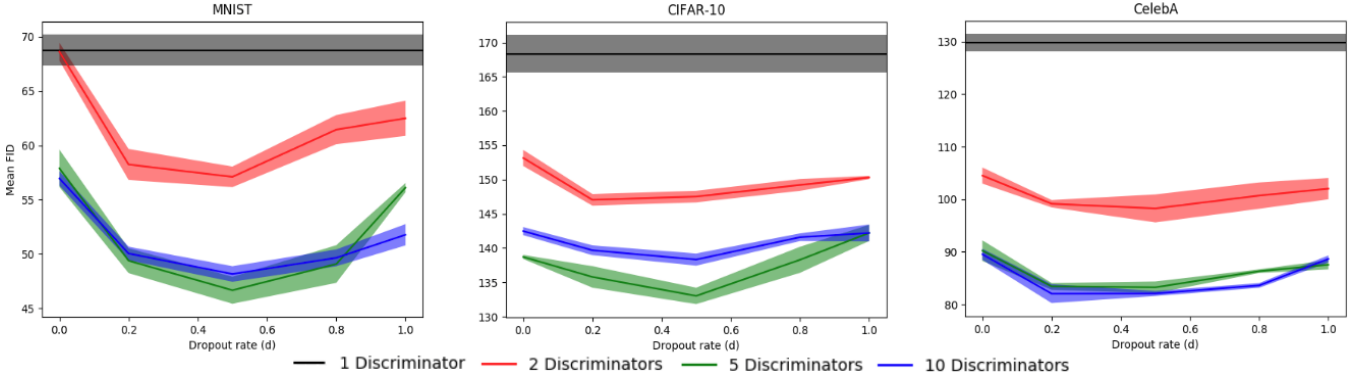
Figure 6: Mean FID calculated across 100 epochs on the different datasets. Smaller values mean better looking and more varied generated samples over time. The convex representation of FID indicates the benefits in using mid-range dropout rates.
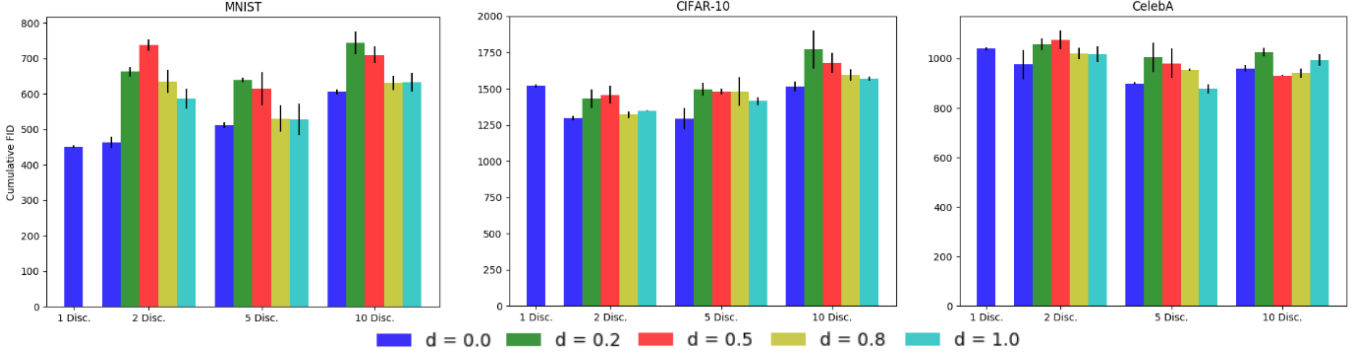

Figure 7: Cumulative FID calculated between every consecutive epoch. Higher values represent more diversity across time.

Table 2: Inception score comparison between Dropout-GAN and different variants of GMAN. Original GANs with the modified loss is also presented as a baseline.

	1 DISC.	2 DISC.	5 DISC.
GANs	5.74 ± 0.17	-	-
GMAN-0	-	<u>5.88 ± 0.19</u>	5.96 ± 0.14
GMAN-1	-	5.77 ± 0.16	<u>6.00 ± 0.19</u>
GMAN*	-	5.54 ± 0.09	5.96 ± 0.15
DROPOUT-GAN ($d = 0.2$)	-	5.95 ± 0.10	6.01 ± 0.12
DROPOUT-GAN ($d = 0.5$)	-	**5.98 ± 0.10**	**6.05 ± 0.15**

are presented in Table 2. Dropout-GAN outperforms both methods for all different number of discriminators scenarios on all tested dropout rates.

8 RELATED WORK

We now focus on analyzing previous studies that aim to mitigate the mode collapse problem with GANs. Instead of extending the original framework to multiple adversaries, one can change GANs objective to directly promote sample diversity. [3, 6, 24] proposed to optimize distance measurements to stabilize training. On the other

hand, [36, 38] reformulated the original GANs problem using an energy-based objective to promote sample variability. [8] makes use of an autoencoder to penalize missing modes and regularize GANs objective, and UnrolledGAN [28] changes G objective to satisfy an unrolled optimization of D.

By not constraining our approach to a specific objective function, the approaches described above can be viewed as complementary work to our solution. Hence, applying the described principles presented in this paper to such frameworks would be a viable step to further promote diversity among the generated samples for all the previously described methods.

While some work has also focused on augmenting the number of generators [13, 14, 19], or even increasing both the number of generators and discriminators [7, 12, 16, 31], we will now turn our focus on methods that solely increases the number of discriminators in order to prevent mode collapse.
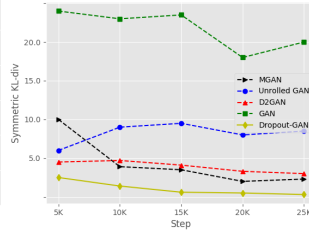
D2GAN [30] proposed a single generator dual discriminator architecture where one D rewards samples coming from the true data distribution, while the other rewards samples that are likely to come from G. Thus, each D still operates on a different objective, similarly to what was previously proposed in [9], where the different objectives might even be conflicting. GMAN [10] proposed a framework where a single G is trained against several discriminators considering different levels of difficulty. This is accomplished by either
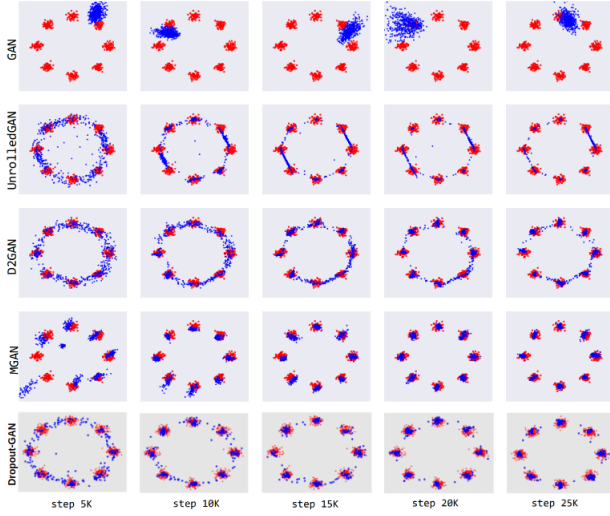
(a) Toy dataset.
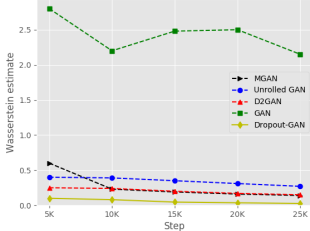
(b) Wasserstein distance.   (c) Symmetric KL divergence.

Figure 8: Comparison of Dropout-GAN (8 discriminators) with original GANs [15], Unrolled GAN [28], D2GAN [30], and MGAN [19] (a). Real data is presented in red while generated data is presented in blue. Our method manages to cover all modes with significantly less noisy samples that fall outside any real mode when compared to the other discriminator modified methods (please note that MGAN uses 8 generators, 1 discriminator and an extra classifier in their framework). Dropout-GAN also achieves the lowest distance (b) and divergence (c) between the real and generated data, continuously converging to 0 over time.

using the average loss of all discriminators or by picking only the *D* with the maximum loss in relation to *G*'s output. [29] proposed to train a single generator against an array of discriminators that operate on a different low-dimensional projection of the data.

However, both of these two last approaches condition the discriminator's architecture to promote variety either by restricting *D*'s architecture to be convolutional or by using different architectures for each *D*. We argue that this is a limitation from an extensibility standpoint, and, therefore, we apply dropout to achieve a similar effect without compromising the extendability of our framework, while continuing to apply the principles of the original GANs objective functions on all of our models.

9   CONCLUSION AND FUTURE WORK

In this work, we propose to mitigate mode collapse by proposing a new framework, called Dropout-GAN, that enables a single generator to learn from an ensemble of discriminators that dynamically changes at the end of every batch. We conducted experiments on multiple datasets that show that Dropout-GAN successfully contributes to a bigger sample variety, both within and across different epochs. Moreover, it also contributes in increasing training stability over time by enabling *G* to receive more quantity and variety of feedback.

In the future, it would be interesting to adjust *G*'s learning rate according to the size of the discriminator set in an inverse manner. This would allow a more coherent learning speed between *G* and the different discriminators, especially when using a large ensemble. Moreover, using game theory techniques to make the different discriminators dependent, *i.e.,* aware of each other's feedback, could also be a very interesting path to follow. This could ultimately optimize our framework, by taking full advantage of using multiple adversarial training.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.
[2] Martin Arjovsky and Leon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR 2017)*.
[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 214–223.
[4] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. Generalization and Equilibrium in Generative Adversarial Nets (GANs). In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 224–232.
[5] Pierre Baldi and Peter J Sadowski. 2013. Understanding Dropout. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2814–2822.
[6] K. Y. Chau Lui, Y. Cao, M. Gazeau, and K. Shuangjian Zhang. 2017. Implicit Manifold Learning on Generative Adversarial Networks. *ArXiv e-prints* (Oct. 2017). arXiv:stat.ML/1710.11260
[7] Tatjana Chavdarova and Francois Fleuret. 2018. SGAN: An Alternative Training of Generative Adversarial Networks. In *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition*.
[8] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. 2016. Mode Regularized Generative Adversarial Networks. *CoRR* abs/1612.02136 (2016). arXiv:1612.02136
[9] Sahil Chopra and Ryan Holmdahl. [n. d.]. Modified-Adversarial-Autoencoding Generator with Multiple Adversaries: Optimizing Multiple Learning Objectives for Image Generation with GANs. ([n. d.]).
[10] Ishan P. Durugkar, Ian Gemp, and Sridhar Mahadevan. 2016. Generative Multi-Adversarial Networks. *CoRR* abs/1611.01673 (2016). arXiv:1611.01673
[11] Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*. 1019–1027.
[12] Zhe Gan, Liqun Chen, Weiyao Wang, Yunchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. 2017. Triangle Generative Adversarial Networks. *CoRR* abs/1709.06548 (2017). arXiv:1709.06548
[13] Arnab Ghosh, Viveka Kulharia, and Vinay P. Namboodiri. 2016. Message Passing Multi-Agent GANs. *CoRR* abs/1612.01294 (2016). arXiv:1612.01294

[14] Arnab Ghosh, Viveka Kulharia, Vinay P. Namboodiri, Philip H. S. Torr, and Puneet Kumar Dokania. 2017. Multi-Agent Diverse Generative Adversarial Networks. *CoRR* abs/1704.02906 (2017). arXiv:1704.02906

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680.

[16] Aditya Grover and Stefano Ermon. 2017. Boosted Generative Models. *CoRR* abs/1702.08484 (2017). arXiv:1702.08484

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 6629–6640.

[18] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580 (2012).

[19] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Q. Phung. 2017. Multi-Generator Generative Adversarial Nets. *CoRR* abs/1708.02556 (2017). arXiv:1708.02556 http://arxiv.org/abs/1708.02556

[20] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 1857–1865.

[21] Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. 2017. How to Train Your DRAGAN. *CoRR* abs/1705.07215 (2017). arXiv:1705.07215

[22] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images.* Technical Report.

[23] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010).

[24] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabas Poczos. 2017. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 2200–2210.

[25] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[26] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2017. Are GANs Created Equal? A Large-Scale Study. *arXiv preprint arXiv:1711.10337* (2017).

[27] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. The Numerics of GANs. In *Proceedings Neural Information Processing Systems*.

[28] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled Generative Adversarial Networks. *CoRR* abs/1611.02163 (2016). arXiv:1611.02163

[29] Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. 2017. Stabilizing GAN Training with Multiple Random Projections. *CoRR* abs/1705.07831 (2017). arXiv:1705.07831

[30] Tu Dinh Nguyen, Trung Le, Hung Vu, and Dinh Q. Phung. 2017. Dual Discriminator Generative Adversarial Nets. *CoRR* abs/1709.03831 (2017). arXiv:1709.03831

[31] Yuxin Peng, Jinwei Qi, and Yuxin Yuan. 2017. CM-GANs: Cross-modal Generative Adversarial Networks for Common Representation Learning. *CoRR* abs/1710.05106 (2017). arXiv:1710.05106 http://arxiv.org/abs/1710.05106

[32] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015). arXiv:1511.06434 http://arxiv.org/abs/1511.06434

[33] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. 2017. Stabilizing Training of Generative Adversarial Networks through Regularization. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 2015–2025.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[36] Calvin Seward GÃijnter Klambauer Martin Heusel Hubert Ramsauer Sepp Hochreiter Thomas Unterthiner, Bernhard Nessler. 2018. Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields. *International Conference on Learning Representations* (2018).

[37] David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. 2013. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197* (2013).

[38] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. 2016. Energy-based Generative Adversarial Network. *CoRR* abs/1609.03126 (2016). arXiv:1609.03126

A ALGORITHM

The overall training procedure of Dropout-GAN is described in Algorithm 1. We assume the number of samples given to each discriminator at each batch to be a multiple of the overall batch size, *i.e.*, m is an integer. K represents the number of discriminators, d the dropout rate, and B the batch size.

Initialize: $m \leftarrow \frac{B}{K}$
for each epoch **do**
  for each batch **do**
    for $k = 1$ to $k = K$ **do**
      • Sample minibatch z_i, $i = 1 \ldots m$, $z_i \sim p_g(z)$
      • Sample minibatch x_i, $i = 1 \ldots m$, $x_i \sim p_r(x)$
      • Update D_k by ascending along its gradient:

$$\nabla_{\theta_{D_k}} \frac{1}{m} \sum_{i=1}^{m} [\log D_k(x_i) + \log(1 - D_k(G(z_i)))]$$

    end for
    • Sample minibatch δ_k, $k = 1 \ldots K$, $\delta_k \sim Bern(1 - d)$
    if $all(\delta_k) = 0$ **then**
      • Sample minibatch z_i, $i = 1 \ldots m$, $z_i \sim p_g(z)$
      • Update G by descending along its gradient from a random discriminator D_j, for some $j \in \{1, ..., K\}$:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D_j(G(z_i)))$$

    else
      • Sample minibatch z_{k_i}, $i = 1 \ldots m$, $k = 1 \ldots K$, $z_{k_i} \sim p_g(z)$
      • Update G by descending along its gradient:

$$\nabla_{\theta_G} \sum_{i=k}^{K} \delta_k (\frac{1}{m} \sum_{i=1}^{m} \log(1 - D_k(G(z_{k_i}))))$$

    end if
  end for
end for

Algorithm 1: Dropout-GAN.