DOG: DISCRIMINATOR-ONLY GENERATION

Franz Rieger[1,2] **& Joergen Kornfeld**[1]
[1] Max Planck Institute for Biological Intelligence, Martinsried, Germany
[2] Technical University of Munich, Munich, Germany
{first.last}@bi.mpg.de

ABSTRACT

As an alternative to generative modeling approaches such as denoising diffusion, energy-based models (EBMs), and generative adversarial networks (GANs), we explore discriminator-only generation (DOG). DOG obtains samples by direct gradient descent on the input of a discriminator. DOG is conceptually simple, generally applicable to many domains, and even trains faster than GANs on the QM9 molecule dataset. While DOG does not reach state-of-the-art quality on image generation tasks, it outperforms recent GAN approaches on several graph generation benchmarks, using only their discriminators.

1 INTRODUCTION

Generative modeling approaches, such as denoising diffusion Sohl-Dickstein et al. (2015); Ho et al. (2020) and GANs Goodfellow et al. (2014); Sauer et al. (2023), have revolutionized content creation in various domains. Given their wide-ranging potential, thoroughly exploring their design space is essential.

GANs consist of a generator, which generates samples from noise vectors to match a real data distribution, and a discriminator, which assigns a realness score to distinguish between real and generated samples. Notably, the discriminator itself is discarded after training, and only the generator remains in use. While optimizing generator architectures has received much attention Karras et al. (2019; 2020; 2021); Walton et al. (2022); De Cao & Kipf (2018); Krawczuk et al. (2021); Martinkus et al. (2022), we explore DOG, an approach that removes the need for a generator altogether and instead directly utilizes the information stored in the discriminator. Conceptually, DOG is a middle ground between GANs and EBMs Ackley et al. (1985); Du & Mordatch (2019), making it a consequential next step in exploring generative modeling approaches.

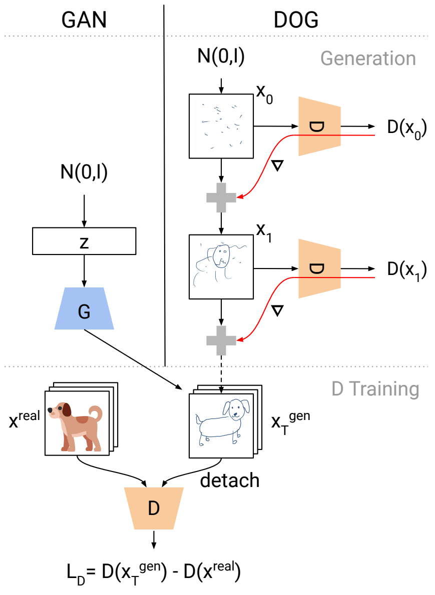


Figure 1: For generation, DOG optimizes a sample directly by gradient descent w.r.t. a generation loss on the output of the discriminator. To train D, sample generation (top) and D updates (bottom) are alternated, like in a GAN, but without requiring a generator.

DOG uses only a single discriminator model for generation. It starts with a pure noise sample in the target domain and optimizes it directly by gradient descent w.r.t. a generation loss on the output of the discriminator. The generated samples are then used, together with real samples, to train the discriminator by alternating sample generation and discriminator update, as shown in Figure 1. In contrast to GANs, where the generator and the discriminator are each other's adversaries, the DOG discriminator acts as its own adversary.

Our main contribution is to demonstrate that DOG is applicable to several domains, such as a 2D-Gaussian toy dataset, the FFHQ face image dataset, and several graph benchmark datasets. Strikingly, DOG outperforms GANs on graphs without adjusting the discriminator architecture or the original GAN hyperparameters.

Algorithm 1 PyTorch-style pseudocode for DOG training.

```
discriminator_opt = Adam(D, learning_rate_d)
for x_real in dataloader: # train D
  x_gen = Parameter(randn_like(x_real))
  x_gen_opt = SGD([x_gen], learning_rate_g)
  D.requires_grad_(False)
  for _ in range(T): # optimize x_gen
    x_gen_opt.zero_grad()
    loss_g = -D(x_gen).mean()
    loss_g.backward()
    x_gen_opt.step()
  D.requires_grad_(True)
  discriminator_opt.zero_grad()
  loss_d = D(x_gen.detach()).mean() \
               - D(x_real).mean()
  loss_d.backward()
  discriminator_opt.step()
```

2 Method

GANs rely on two loss functions: L_G, which incentivizes the generator G to generate samples x^{gen} that receive high scores from the discriminator D, and L_D, which incentivizes the discriminator D to assign low scores to x^{gen} and high scores to real samples x^{real}. A popular example are Wasserstein losses Arjovsky et al. (2017):

$$L_G = -D(x^{\text{gen}}) \tag{1}$$

$$L_D = D(x^{\text{gen}}) - D(x^{\text{real}}) \tag{2}$$

For DOG, we reuse these loss functions but replace the sample generation via a generator G with a generation optimization (GO) process. GO aims to generate samples x^{gen} that (locally) minimize L_G, similar to samples from a GAN generator G. Since there is no generator model G in DOG, we refer to L_G as a "generation" loss instead of a "generator" loss. GO depends on the current discriminator D and a random starting sample $x_0^{\text{gen}} \sim P = N(0, I)$. We generate a sample $x^{\text{gen}} = x_T^{\text{gen}} = \text{GO}(D, x_0^{\text{gen}})$ by iteratively updating the sample using the gradient of L_G with respect to x_t^{gen}. The sequence of samples x_t^{gen} forms a GO path. We optimize for T steps using gradient descent with a learning rate η:

$$x_{t+1}^{\text{gen}} = x_t^{\text{gen}} - \eta \nabla_{x_t^{\text{gen}}} L_G(x_t^{\text{gen}}) \tag{3}$$

We train the discriminator model D to minimize the discriminator loss $L_D(x^{\text{real}}, \text{sg}(x^{\text{gen}}))$, where sg is a stop gradient operator. As described in Algorithm 1, this process is repeated for each training step.

We use the stop gradient operator sg to prevent collapse: Suppose we kept the gradient from GO for the discriminator weight update step, and also required gradient computation for the weights of D. Then, the computational graph through which we backpropagate for a training step would contain all the T forward and backward passes through D from GO, which would be problematic due to memory constraints. In addition, since L_D pushes D to assign a lower score to the final generated sample x_T^{gen}, D would have an advantage in modifying its score surface so that there are no GO paths from noise to realistic samples, e.g., by giving zero gradient for noisy inputs.

DOG is flexible in the choice of P, L_G, and L_D. In practice, we also use learning rate scheduling, so η depends on t, and Equation (3) is extended with the update rules of momentum-based optimizers. For a formal analysis of DOG, see Appendix C.

3 Evaluation

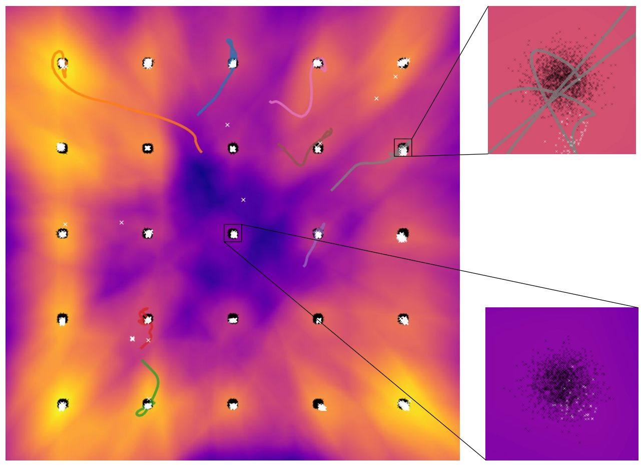
To evaluate the versatility of DOG, we tested it on various datasets across multiple domains. These datasets include a 2D toy dataset with 25 Gaussians Tran et al. (2018), FFHQ, a dataset of face images Karras et al. (2019), and several graph datasets such as QM9, a molecule dataset Wu et al. (2018).

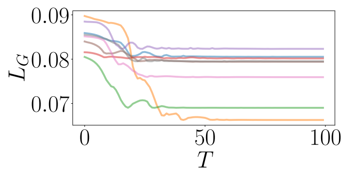
Based on the results of early experiments, we generally use Adam Kingma & Ba (2015) as the optimizer for the GO process with $T = 100$ and a OneCycle learning rate schedule Smith & Topin (2019) (max_lr $= 1.0$ and a warm-up ratio of 0.3) for faster convergence than constant learning rate gradient descent. An implementation can be found at `https://github.com/riegerfr/dog`.

3.1 25-GAUSSIANS TOY DATASET

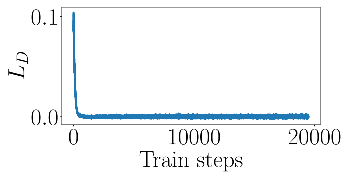
To generate a 25-Gaussians dataset, we independently draw 2000 samples from a Gaussian with covariance $\Sigma = I * 0.0001$ for each mean μ in $[-1.5, -0.75, 0, 0.75, 1.5]^2$ to obtain $50,000$ samples, similar to Tran et al. (2018). As in Wang et al. (2022), we use a discriminator with 2 hidden linear layers of 128 units each and LeakyReLU activations between them. We set max_lr $= 0.1$ for the OneCycle in the GO process and train with a batch size of 128 for 50 epochs using Adam with lr $= 10^{-5}$, $(\beta_1, \beta_2) = (0, 0.9)$, and Wasserstein losses Arjovsky et al. (2017).

As shown in Figure 2, all modes of the data are covered, and most of the generated samples are close to the real modes. Note that it is possible for the discriminator scores (and the resulting loss surface) to be not perfectly symmetric and for some modes to receive higher scores than others, while still providing many GO paths to each mode. In particular, the central mode at $(0, 0)$ is still covered even though it receives a lower score than others since GO often ends at local maxima. Furthermore, since the starting point density is $N(0, I)$, which is highest at $(0, 0)$, we already initialize many GO paths close to this mode.


(a) Discriminator surface



(b) Generation loss



(c) Training loss

Figure 2: Results of DOG on the 25-Gaussians toy dataset. (a) The background colors represent the discriminator scores, with brighter colors indicating higher scores. The black crosses represent the $50,000$ real samples, while the white crosses represent $1,280$ generated samples. The colored lines show a subset of the GO paths, which start at random locations x_0^{gen} and end at the generated samples x_T^{gen}. (b) Typical GO loss curves, colored according to the paths shown in (a). Both the initial and final scores vary, and all GOs have converged. (c) Discriminator training loss of 25-Gaussians for DOG. A loss close to zero indicates that generated and real samples receive similar scores.

3.2 FACE IMAGE DATA

For a first demonstration of DOG on image data, we utilize FFHQ (256² pixels) Karras et al. (2019) and follow the settings of StyleNAT Walton et al. (2022) to a large extent. However, due to the high computational cost of DOG, we use a slimmer discriminator architecture with a quarter of the channels and train the model for only 100 epochs, which is about an order of magnitude less than StyleNAT. Other hyperparameters and regularization terms, such as non-saturating (NS) GAN losses, horizontal flips for data augmentation, balanced consistency regularization (bCR) Zhao et al. (2021), R1 regularization Mescheder et al. (2018), a batch size of 64, and Adam with lr $= 2 * 10^{-4}$ and $(\beta_1, \beta_2) = (0, 0.9)$, remain the same. While an exponential moving average was used for the generator in the original setting, we do not apply it to the discriminator for DOG.

Although the displayed images in Figure 3 show faces, their quality is not competitive with the current state of the art, as quantified by a FID-50k of 142.86 for DOG versus 2.05 for StyleNAT. Since discriminators have only been used in conjunction with generators so far, they may implicitly exploit the generator's architecture. Thus, to enable better performance for image generation, DOG may require substantial changes to the discriminator architecture, larger models, different regularization techniques, and tuning of hyperparameters. This should be explored in future work, after finding ways to reduce the computational cost of DOG, as discussed in Appendix B.

3.3 Graph Data

To generate graphs using DOG, we directly optimize both the node feature matrix and the edge feature matrix (or adjacency matrix if the edges have no features). At each generation step in GO, we apply constraints to the edge matrix by setting its diagonals to zero and ensuring symmetry. Moreover, we mask the edge matrix and the node feature matrix based on the given number of nodes for each batch element, similar to the post-processing step used by Martinkus et al. (2022) for SPECTRE, which is the current state-of-the-art for graph generation using GANs.

Data For a detailed description of each dataset used for the graph generation evaluation, data splits, and statistics, we refer to Martinkus et al. (2022). In short, we evaluate DOG on benchmark datasets that include both artificial and real-world graphs. The artificial datasets are Community-small You et al. (2018), with 12-20 nodes; SBM, which includes graphs sampled from stochastic block models; and Planar, containing only planar graphs. The two real-world datasets are Proteins Dobson & Doig (2003), containing proteins with hundreds of nodes, and QM9 Ramakrishnan et al. (2014), an exhaustive dataset of molecule graphs with up to 9 heavy atoms.

Metrics Following Martinkus et al. (2022), we generate as many graphs for each dataset as there are in the test set. For the set of generated graphs, we report the percentage of valid, unique, and novel (V., U. & N.) graphs, as well as the maximum mean discrepancy (MMD) for various statistics (Degree, Clustering, Orbit, Spectrum, and Wavelet) between the generated set and the test set, where applicable. We also report the average ratio among all statistics between the MMD of the generated set and the MMD of the training set, where a ratio of 1.0 indicates that the generated set is as distinguishable from the test set as the training set. For Community-small, we use the earth mover's distance (EMD) instead of MMD, to maintain consistency with Martinkus et al. (2022).

Baselines As baselines, we compare the results of DOG with those of SPECTRE and GG-GAN (RS)*, as reported in Martinkus et al. (2022). Note that they use the node number distribution of the test set to generate samples, which we also adopt for consistency. Additionally, we include GG-GAN* or MolGAN* Martinkus et al. (2022) where they perform better. Where available, we also include the results of DiGress Vignac et al. (2023), a recent diffusion-based approach that does not use a discriminator and updates the graph in discrete steps. Note that Vignac et al. (2023) report the MMD ratios directly instead of the raw MMD values. Therefore, we calculate them using the training MMD values of Martinkus et al. (2022), which may introduce rounding errors. The comparability of the results is further limited because DiGress uses different splits for some datasets and uses the node distribution of the training set for sampling.

Model Although DOG could use the more advanced discriminator of SPECTRE, which uses conditioning on eigenvalues and eigenvectors, we focus on the simple discriminator architecture of GG-GAN (RS)* with default hyperparameters, following the reference GG-GAN (RS)* implementation also introduced by Martinkus et al. (2022). For example, for QM9, we use a discriminator with 3 PPGN layers Maron et al. (2019) with 64 channels each and a batch size of 128.

Other Settings Like Martinkus et al. (2022), we train for 30 epochs for QM9 and $12,000$ epochs for Community-small and Planar. Due to the expensive GO, we use only 130 epochs for Proteins (instead of $1,020$) and $2,400$ for SBM (instead of $6,000$). Each run uses a single Nvidia A40, except for Proteins where we use 2 GPUs (and maintain a total batch size of 4 as in SPECTRE). Training on our default QM9 configuration takes about 15 hours. We keep the seeds constant as in the reference implementation and, like Martinkus et al. (2022), select a checkpoint for testing according to the validation performance. As the validation metric, we use the mean MMD ratio, except for QM9, where we choose the ratio of valid and unique generated graphs. Like Martinkus et al. (2022), we use WGAN-LP ($\lambda_{LP} = 5$) and gradient penalty as losses Petzka et al. (2018). We train using Adam with lr $= 10^{-4}$ and $(\beta_1, \beta_2) = (0.5, 0.9)$.

QM9 Results As shown in Table 1, DOG outperforms all GAN approaches on QM9, including GG-GAN (RS)* and SPECTRE, while being competitive with DiGress for a larger discriminator (6 layers, 128 channels each). For our main experiment with the default configuration, we report the mean and standard deviation for five seeds. See Figure 4 for examples of generated graphs.

Table 1: QM9 results. °As discussed by Vignac et al. (2023) novelty is a problematic metric for QM9. Therefore, we report it here only for completeness.

METHOD	V.↑	V. & U. ↑	V., U.& N.°
DIGRESS	**99.0**	≈<u>95.2</u>	-
SPECTRE	87.3	31.2	29.1
GG-GAN (RS)*	51.2	24.4	24.4
DOG 3L. 64 CH.	93.8±1.3	91.7±0.9	58.1±2.4
DOG 6L. 128 CH.	<u>98.9</u>	**95.8**	42.0

Hyperparameter Study We conducted a hyperparameter study on QM9 to evaluate the impact of our design choices on the performance of our model. In Table 2, we start with the default model and examine the effect of varying hyperparameters on the fraction of valid and unique graphs. We also analyze the discriminator's (smoothed) Wasserstein loss L_D at the end of training since we observed in early experiments that a low loss correlates with poor sample quality, suggesting that it is not desirable for GO if the discriminator can distinguish well between generated and real samples.

First, we investigate the model size by doubling the number of channels and layers, both independently and together. We find that models larger than those used by the GAN baselines perform better.

Table 2: Hyperparameter study for DOG on QM9. Default: 100 OneCycle Adam steps, 3 layer, 64 channels, WGAN-LP losses. °Not comparable because different loss functions or number of epochs are used.

SETTING	V & U. ↑	FINAL L_D
DEFAULT DOG	91.7±0.9	-0.17±0.04
128 CH.	93.8	-0.08
6 LAY.	<u>94.3</u>	-0.21
128 CH., 6 LAY.	**95.8**	-0.14
200 STEPS	92.0	-0.07
50 STEPS	81.3	-0.79
25 STEPS	34.1	-44
MAX_LR = 0.1	90.2	-0.29
ADAM → SGD	91.1	-0.16
NO ONECYCLE	56.9	-15
NO CONSTRAIN	81.8	-0.34
NS LOSSES	84.4	1.27°
1 EPOCH TRAINING	43.4	-0.04°

Next, we examine our choices for GO: While increasing the number of steps T may result in better convergence to local minima of L_G (maxima in D's score surface), it also increases the computational cost of DOG, posing a trade-off. Doubling the number of steps to $T = 200$ results in higher discriminator losses but similar quality, indicating that $T = 100$ is sufficient. In contrast, halving the number of steps to $T = 50$ leads to worse performance. An even lower number of steps ($T = 25$) leads to a failure mode where GO ends before the discriminator can be fooled, giving the discriminator a low loss. Reducing the learning rate in GO by a factor of 10 leads to only slightly worse results, indicating that GO is not overly sensitive to learning rates, at least when using Adam with OneCycle learning rate scheduling as the optimizer. However, not using learning rate scheduling (but tuned lr $= 0.1$) significantly reduces the sample quality, suggesting that learning rate scheduling is crucial for GO, as again implied by the low discriminator loss. Replacing Adam with stochastic gradient descent (SGD) in the GO process performs similarly to the default setting but requires more learning rate tuning (max_lr $= 100$). Not constraining the edge matrix after each GO step has a negative effect, demonstrating the benefit of staying within the target domain during GO.

Notably, replacing the WGAN losses with NS losses (but keeping λ_{LP} and other hyperparameters) leads to only a moderate degradation, supporting our claim that DOG is flexible regarding the choice of L_D and L_G.

While individual training steps of DOG take more time due to the iterative GO, we find that the overall training nevertheless progresses faster compared to SPECTRE. Although SPECTRE trains by default for 30 epochs ($\approx$ 8 hours) on QM9, DOG achieves already better test performance after only one epoch ($<$ 0.5 hours), indicating faster convergence. We speculate that GO yields higher quality generated samples for the discriminator early on: Unlike a generator model in a GAN, GO does not have parameters that are adopted to the discriminator weights with a delay. This observation suggests that in DOG, inner (generation) and outer (training) optimization are closely related.

Other Graph Datasets The results for other graph datasets also demonstrate DOG's superior graph generation performance compared to the baseline GG-GAN (RS)* and SPECTRE. This trend

Table 4: Planar (top) and SBM (bottom) results.

Method	Deg. ↓	Clus. ↓	Orbit ↓	Spec. ↓	Wavelet ↓	Ratio ↓	V. ↑	U. ↑	N. ↑	V., U. & N. ↑
DiGress	≈0.00028	≈**0.0372**	≈**0.00085**	-	-	-	-	-	-	**75**
SPECTRE	0.0005	0.0785	0.0012	0.0112	0.0059	2.9	25.0	**100.0**	**100.0**	25.0
GG-GAN (RS)*	0.1005	0.2571	1.0313	0.2040	0.3829	586.3	0.0	**100.0**	**100.0**	0.0
DOG (ours)	**0.00023**	0.0827	0.0034	**0.0067**	**0.0029**	**2.78**	**67.5**	**100.0**	**100.0**	67.5

Method	Deg. ↓	Clus. ↓	Orbit ↓	Spec. ↓	Wavelet ↓	Ratio ↓	V. ↑	U. ↑	N. ↑	V., U. & N. ↑
DiGress	≈0.00128	≈**0.0498**	≈0.04335	-	-	-	-	-	-	**74**
SPECTRE	0.0015	0.0521	0.0412	0.0056	0.0028	2.0	52.5	**100.0**	**100.0**	52.5
GG-GAN (RS)*	0.0338	0.0581	0.1019	0.0613	0.1749	61.5	0.0	**100.0**	**100.0**	0.0
GG-GAN*	0.0035	0.0699	0.0587	0.0094	0.0202	7.8	25.0	**100.0**	**100.0**	25.0
DOG (ours)	**0.0003**	0.0508	**0.0401**	**0.0039**	**0.0013**	**1.15**	**72.5**	**100.0**	**100.0**	72.5

Table 5: Proteins results. °Note that novelty is severely limited for MolGAN* as discussed by Martinkus et al. (2022).

Method	Deg. ↓	Clus. ↓	Orbit ↓	Spec. ↓	Wavelet ↓	Ratio ↓	U. ↑	N. ↑	U. & N. ↑
SPECTRE	0.0056	0.0843	0.0267	0.0052	0.0118	16.9	**100.0**	**100.0**	**100.0**
GG-GAN (RS)*	0.4727	0.1772	0.7326	0.4102	0.6278	875.8	**100.0**	**100.0**	**100.0**
MolGAN*	**0.0008**	**0.0644**	**0.0081**	**0.0021**	**0.0012**	**4.2**	97.3	**100.0**°	97.3
DOG (ours)	0.0022	0.0682	0.0202	0.0014	0.0023	6.75	**100.0**	**100.0**	**100.0**

can be observed in Tables 3 to 5, which present the results for Community-small, Planar, SBM (max_lr $= 0.1$), and Proteins (max_lr $= 0.1$), respectively. Many of these results are on par with DiGress. Notably, DOG achieves this high performance using only the simple GG-GAN (RS)* discriminator, and we did not tune any GAN-specific hyperparameters. We only tuned the DOG-specific max_lr and kept $T = 100$ constant. For examples of both real and generated graphs, see Figure 5.

4 Related work

Prior work that is conceptually similar to DOG are EBMs Ackley et al. (1985); Du & Mordatch (2019). EBMs train an energy estimation model to assign a high energy score to real samples and a low energy score to generated samples, essentially using a Wasserstein loss Arjovsky et al. (2017). This energy defines a density from which one can sample using Langevin dynamics Welling & Teh (2011). Langevin dynamics use noise and the gradient of the energy score with respect to the input sample to update the input sample in several steps. A key difference between DOG and EBMs is that DOG does not use noise during GO, but only deterministic gradient descent w.r.t. an explicit generation loss L_G, possibly paired with learning rate scheduling and momentum-based optimization. DOG's sample density is higher at local maxima in the discriminator's score surface (minima of L_G), as GO paths ideally end there. Since a high gradient at a location usually causes the gradient-only GO to move away from it, a high score from a DOG discriminator does not necessarily define a high density. Thus, conceptually, the model in EBMs is an energy estimator that defines a (unnormalized) density, while in DOG, it is a discriminator that should distinguish between generated and real samples. Another difference between DOG and EBMs is the lack of a sample replay buffer, which EBMs use during training to save expensive generation steps.

Table 3: Community-small results. We explain the extraordinarily small ratio of EMD scores by the small number of test samples (and thus generated samples), as well as the fact that we use the node distribution of the test set to be consistent with Martinkus et al. (2022). Our results, as well as those of DiGress indicate that the performance is saturated for this toy dataset.

Method	Deg. ↓	Clus. ↓	Orbit ↓	Ratio ↓
DiGress	≈0.018	≈0.0643	≈0.006	1.0
SPECTRE	0.02	0.21	0.01	1.7
GG-GAN (RS)*	0.08	0.22	0.08	5.5
DOG (ours)	**0.003**	**0.006**	**0.002**	**0.16**

Other approaches related to EBMs and DOG include score-based generative modeling Song & Ermon (2019); Song et al. (2021) and denoising diffusion approaches Sohl-Dickstein et al. (2015); Ho et al. (2020). All of these methods generate samples incrementally by making dense predictions on the input, starting from pure noise. They allow for corrections of previous inaccuracies during generation, unlike GANs, which generally generate samples in a one-shot fashion. Diffusion models typically predict the remaining noise, while score-based models estimate the gradient of the energy surface. Unlike for EBMs and DOG, the dense prediction in these methods does not come from a backward pass through a scalar prediction model but from a forward pass through a dense prediction model. In addition, the settings do not require expensive sample generation during training.

Other approaches, such as DiffusionGANs Wang et al. (2022) and discriminator guidance Kim et al. (2022), combine ideas from diffusion and GANs by using a discriminator to refine partially denoised samples generated by a diffusion model. EBMs and GANs are also related. While combinations were proposed early on Zhao et al. (2017), Che et al. (2020) show how a GAN can be interpreted as an EBM, and Xiao et al. (2021) indicate that EBMs are self-adversarial like DOG. Additionally, Ansari et al. (2020) show how to use the gradient of a discriminator to refine the generation, which may come from a GAN generator. Besides these, a range of other popular generative approaches have been explored, including normalizing flows Rezende & Mohamed (2015); Kobyzev et al. (2019), variational autoencoders Kingma & Welling (2014); Vahdat & Kautz (2020), autoregressive generation Brown et al. (2020); Lee et al. (2022), and masked modeling Chang et al. (2022; 2023).

DOG is also related to adversarial robustness Song et al. (2018); Dong et al. (2020): Both settings use the gradient of a model's input to perturb samples, resulting in a change of the model's predictions. However, the goal of DOG is different: it aims at de novo content generation and receives only noise as an input during inference, no adversarially perturbed samples.

Specifically for the graph domain, besides the multi-step GAN-based SPECTRE Martinkus et al. (2022) and the state-of-the-art diffusion-based DiGress Vignac et al. (2023), other methods have been proposed. These include score-based generative modeling Niu et al. (2020); Jo et al. (2022) and autoregressive models Liao et al. (2019). Earlier graph GANs are based on recursive neural networks You et al. (2018) or random walks Bojchevski et al. (2018). Other approaches utilize normalizing flows Madhawa et al. (2020) or focus on permutation invariance Vignac & Frossard (2022). Note that all of these approaches are tailored to graphs, sometimes even limited to molecule graphs. In contrast, DOG is a general approach that can be applied to various domains, including graphs.

5 Discussion

5.1 Training Efficiency

While DOG already shows advantages over GANs in certain domains such as graphs, its current limitations are apparent for images (Figure 3). This could be due to the expensive training steps that include GO. A detailed analysis of the computational cost can be found in Appendix D.

A potential solution to improve training efficiency is to reuse generated samples during training: by using a sample replay buffer, such as Du & Mordatch (2019), the starting point for GO could be closer to realistic samples, thus requiring fewer GO steps to achieve convergence. Another solution would be to use the intermediate samples and their gradients, instead of discarding them altogether with the stop gradient operator, to regularize and potentially shorten the GO paths. Additionally, we expect that more hyperparameter tuning for Adam and OneCycle, or using a more suitable GO approach, could further reduce T. By analogy, note that earlier denoising diffusion approaches used thousands of steps, which could be reduced to only a handful by skipping steps and increasing the step size Meng et al. (2022). Another potential way to speed up training and inference would be to operate in latent space instead of image space Rombach et al. (2022).

5.2 Implications of a Missing Generator

While DOG is seemingly parameter-efficient compared to GANs since it does not require the parameters of a generator, the DOG discriminator has a different task to solve. Note that GO can start anywhere and, assuming convergence, will eventually also reach all local maxima in the score surface of D. Thus, a good discriminator should have local maxima only at realistic samples. This observation suggests that the score surface of a DOG discriminator must give meaningful gradients over the entire target domain for the GO path from each noisy sample to a realistic one. This is in contrast to the score surface of a GAN discriminator, which only needs to provide meaningful gradients on the subset of the domain currently covered by the generator. For example, for FFHQ, after teaching the generator the global structure of a face, the GAN discriminator could adapt to focus on the local texture. Therefore, we speculate that a DOG discriminator might benefit from an architecture and regularization techniques that are optimized to accommodate this difference.

5.3 Conceptual Advantages

One of the primary conceptual advantages of DOG is its ability to instantly update GO with D, providing high-quality training samples to D directly. This is in contrast to GANs, where the generator model must first learn from the discriminator and always lags behind. As a result, DOG may require fewer training steps to achieve good performance, as demonstrated on QM9 (Section 3.3).

In the graph domain, DOG outperforms approaches like SPECTRE Martinkus et al. (2022) by not requiring a complicated multi-step approach where some generators and discriminators need to be trained before others. This greatly simplifies the setting by using only a single discriminator model. Moreover, unlike the extensive journey of different generators from GraphGAN Wang et al. (2018) to SPECTRE, there is no need to tune the generator architecture for DOG. However, the tuning of the discriminator architecture and regularization techniques remains.

Overall, DOG's ability to provide high-quality training samples to D directly and its simplified setting without the need to tune the generator architecture make it a promising generative approach.

6 Conclusion

The self-adversarial DOG shows that high-quality sample generation is possible with a discriminator-only approach on graphs, outperforming GANs while being on par with state-of-the-art diffusion approaches. DOG is conceptually simple and not limited to a specific domain. Making DOG faster and further tuning of the discriminator architecture as well as regularization techniques may also enable competitive performance in other domains.

Acknowledgements

We thank Dominik Fuchsgruber, Anushka Vashishtha, and Daniel Scholz for their valuable feedback and discussions, as well as the anonymous reviewers whose reviews greatly improved the quality of the paper. We are also grateful to the Max Planck Society for generous funding and to Lorenz Hüdepohl and his team at the MPCDF computing facility in Garching for outstanding support.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

David Alvarez-Melis, Vikas Garg, and Adam Tauman Kalai. Why gans are overkill for nlp. *arXiv preprint arXiv:2205.09838*, 2022.

Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. In *International Conference on Learning Representations*, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International conference on machine learning*, pp. 610–619. PMLR, 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1xsqj09Fm`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287, 2020.

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 321–331, 2020.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, 2022.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

Dongjun Kim, Yeongmin Kim, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. *ArXiv*, abs/2211.17091, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

Ivan Kobyzev, Simon Prince, and Marcus A. Brubaker. Normalizing flows: Introduction and ideas. *ArXiv*, abs/1908.09257, 2019.

Igor Krawczuk, Pedro Abranches, Andreas Loukas, and Volkan Cevher. {GG}-{gan}: A geometric graph generative adversarial network, 2021. URL `https://openreview.net/forum?id=qiAxL3Xqx1o`.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Kaushalya Madhawa, Katsuhiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graph{nvp}: an invertible flow-based model for generating molecular graphs, 2020. URL `https://openreview.net/forum?id=ryxQ6T4YwB`.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Karolis Martinkus, Andreas Loukas, Nathanael Perraudin, and Roger Wattenhofer. Spectre : Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, 2022.

Chenlin Meng, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *ArXiv*, abs/2210.03142, 2022.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pp. 3481–3490. PMLR, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1QRgziT-`.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 4474–4484. PMLR, 2020.

Henning Petzka, Asja Fischer, and Denis Lukovnikov. On the regularization of wasserstein GANs. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1hYRMbCW`.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis, 2023. URL `https://arxiv.org/abs/2301.09515`.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pp. 369–386. SPIE, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *Advances in Neural Information Processing Systems*, 31, 2018.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

Ngoc-Trung Tran, Tuan-Anh Bui, and Ngai-Man Cheung. Dist-gan: An improved gan using distance constraints. In *European Conference on Computer Vision*, 2018.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

Clement Vignac and Pascal Frossard. Top-n: Equivariant set and graph generation without exchangeability. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=-Gk_IPJWvk`.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=UaAD-Nu86WX`.

Steven Walton, Ali Hassani, Xingqian Xu, Zhangyang Wang, and Humphrey Shi. Stylenat: Giving each head a new perspective. *arXiv preprint arXiv:2211.05770*, 2022.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Zhisheng Xiao, Qing Yan, and Yali Amit. EBMs trained with maximum likelihood are generator models trained with a self-adverserial loss. In *Energy Based Models Workshop - ICLR 2021*, 2021. URL `https://openreview.net/forum?id=osPDkDQXgI7`.

Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Skj8Kag0Z`.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Pengchuan Zhang, Qiang Liu, Dengyong Zhou, Tao Xu, and Xiaodong He. On the discrimination-generalization tradeoff in GANs. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Hk9Xc_lR-`.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=ryh9pmcee`.

Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved consistency regularization for gans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11033–11041, 2021.

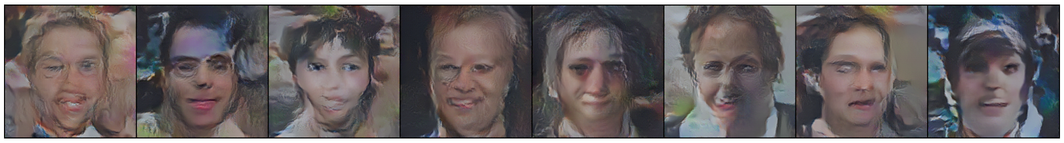


Figure 3: Generated samples for FFHQ using a DOG-trained discriminator.

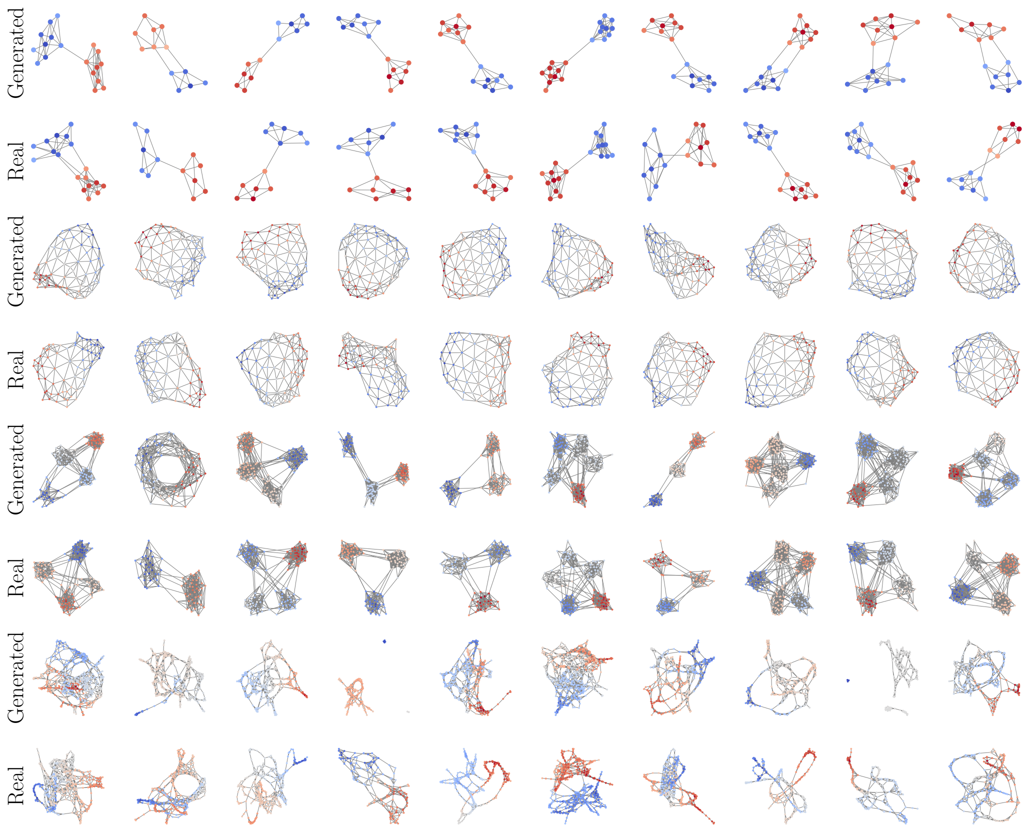


Figure 5: Uncurated set of DOG generated and real samples for Community-small, Planar, SBM and Proteins (from top to bottom). Many properties of the real data are covered by the generated samples, but for example for Proteins, disconnected components can occur.

A Limitations & Societal Impact

DOG harbors the same dangers of deception as other generative approaches such as GANs or denoising diffusion approaches. However, the quality of the generated samples is currently still limited in the image domain and on par with other approaches in the graph domain. Furthermore, DOG training is computationally expensive compared to competitive approaches.

B Image Domain Discussion

We acknowledge that the quality of images generated by DOG is currently low, highlighting the need for different optimizations in the graph and image domains.

It is important to note the significant progress that has been made in generative modeling over the years, with numerous innovations required from early GANs Goodfellow et al. (2014) to GigaGAN Kang et al. (2023), and from early diffusion models Sohl-Dickstein et al. (2015) to simple diffusion Hoogeboom et al. (2023). Given the heavily studied field of image generation, we would be surprised to see near state-of-the-art results in the first paper exploring an innovative generative modeling approach. We argue that extensive hyperparameter tuning, including architecture and regularization techniques, may be required for DOG. For this demonstration, we have only used StyleNAT's GAN-specific hyperparameters without modification and a slim version of the discriminator architecture.

Although some approaches developed for image GANs may be transferable to DOG, an exhaustive evaluation of them is beyond the scope of this paper. We believe that this should be addressed after making DOG faster by finding ways to reduce the number of GO steps T. While most generative approaches are general in theory, some are better suited for specific domains than others. For example, GANs work well for images Kang et al. (2023), but currently not for text Alvarez-Melis et al. (2022). Therefore, we believe that showcasing what works well with DOG (graph generation) and what doesn't work well (image generation) strengthens - not weakens - the paper.

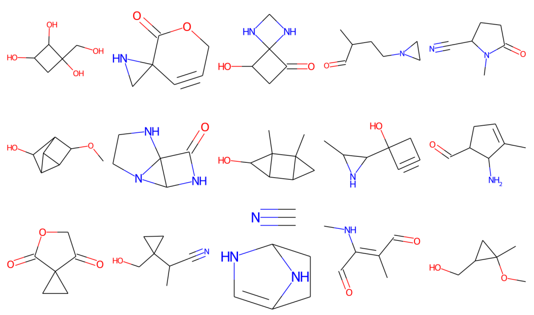


Figure 4: Uncurated set of DOG generated samples for QM9. Like Martinkus et al. (2022), we also count disconnected graphs (like the bottom middle one) as valid for QM9.

Similar to early visualization techniques Mordvintsev et al. (2015), we establish a baseline for DOG's performance by using a discriminator that was trained in the default StyleNAT (GAN) setting without any further training for sampling with DOG's GO without constraints. The results (Adam+OneCycle, max_lr $= 1.0$, $T = 100$) can be observed in Figure 6 (other max_lr lead to similar results). We see that a GAN-trained discriminator does not produce meaningful images, indicating that DOG's gradient-based GO struggles to find a path to a realistic sample when using a GAN-trained discriminator. However, this result is intriguing because it suggests that the DOG training process leads to a fundamentally different discriminator that enables GO paths to more realistic samples as seen in Figure 3.

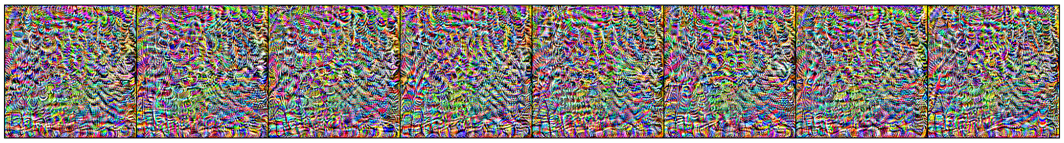


Figure 6: Generated samples for FFHQ using a (fixed) GAN-trained discriminator.

Currently, DOG outperforms GANs on graphs, but not on images. We offer two reasons for this. First, graph discriminators are much lighter than image discriminators, which means that we are not computationally limited by slow training steps. In fact, we trained DOG for the same number of steps, sometimes even using larger discriminators than the best graph GAN (SPECTRE) for QM9. However, for FFHQ, we used an order of magnitude fewer training steps and a much smaller discriminator than the best GAN, as we trained on a single Nvidia A40 for about a week. The second reason is that, given our assumptions, the GO process prefers to reach strict local maxima. Therefore, DOG may be better suited for generating discrete data (such as graphs where there is either an edge or none between two nodes) rather than continuous data such as images. There, a small amount of noise ϵ added to an image x would still be considered a valid sample, whereas DOG would only return $x + \epsilon$ for a fixed subset of ϵ, where $x + \epsilon$ is a local maximum in the discriminator's score surface. To address this limitation, there are two potential remedies that can be explored in future work. One option is to use inference-time dropout in D to create different local maxima for each GO. Another approach is to model pixel values as discrete rather than continuous.

Overall, we believe that DOG's strengths in graph generation showcase the potential of this innovative generative modeling approach, and we hope that our discussion of its current limitations will inspire further research and development in this area.

C Method Analysis

This section provides theoretical insights into DOG's workings. While we utilize Wasserstein GAN loss terms Arjovsky et al. (2017) for simplicity, similar derivations are also applicable to other terms.

For Wasserstein losses, DOG's update rule (Equation (3)) can be simplified to:

$$x_{t+1}^{\text{gen}} = x_t^{\text{gen}} + \eta \nabla_{x_t^{\text{gen}}} D(x_t^{\text{gen}}) \tag{4}$$

C.1 Min-Max Optimization

To better understand DOG from the perspective of saddle-point min-max optimization Yadav et al. (2018), we again consider the Wasserstein GAN. The optimization problem is defined with ψ and θ representing the generator and discriminator parameters respectively, and Z and R representing the latent and real data distribution respectively:

$$\text{min}_{\psi}\text{max}_{\theta} E_{x\sim R}[D_\theta(x)] - E_{z\sim Z}[D_\theta(G_\psi(z))] \tag{5}$$

We could replace G with a perfect adversary of D, which always finds a global maximum in the score surface of D (global minimum for L_G). This adversary thus acts as a better G, knowing directly everything that D has learned:

$$\text{max}_{\theta} E_{x\sim R}[D_\theta(x)] - D_\theta(\text{argmax}_x D_\theta(x)) \tag{6}$$

Since perfect adversaries are not practical, we use a GO to approximate $\text{argmax}_x D_\theta(x)$:

$$\text{max}_{\theta} E_{x\sim R}[D_\theta(x)] - E_{x_0^{\text{gen}}\sim P}[D_\theta(\text{sg}(\text{GO}(D_\theta, x_0^{\text{gen}})))] \tag{7}$$

In practice, we alternate between obtaining a $x^{\text{gen}} = \text{GO}(D_\theta, x_0^{\text{gen}})$ and optimizing θ with a gradient descent step. As described in Section 2, during training, we must prevent gradient computation through GO to ensure that GO remains a good approximator for the argmax.

C.2 Convergence Analysis

For additional insight, we provide a convergence analysis of DOG, analogous to similar analyses for GANs Mescheder et al. (2018), for a simple 1-D dataset and a simple D with only a single (strict) local maximum. Consider Wasserstein losses and the following data distribution: $p(x) = \delta(x)$ (where δ is the Dirac delta function), i.e., the samples are only at $x^{\text{real}} = 0$. Furthermore, consider the following discriminator with a single learnable scalar parameter θ: $D_\theta(x) = -(x-\theta)^2$.

Suppose DOG's T is large enough and the learning rate η in GO is small enough. Then we always converge to the local (and global) maximum in the score surface of the discriminator (minimum for L_G), i.e., $x^{\text{gen}} = \text{GO}(D_\theta, x_0^{\text{gen}}) = \theta + \epsilon$ (with ϵ arbitrarily close to 0 and numerically equal to 0 for all practical purposes).

Then we have:

$$L_D = D_\theta(x^{\text{gen}}) - D_\theta(x^{\text{real}}) = D_\theta(\theta) - D_\theta(0) = 0 + \theta^2 = \theta^2 \tag{8}$$

This is minimized by training with gradient descent using a learning rate $\nu : \theta' = \theta - \nu dL/d\theta = \theta - \nu 2\theta$, which converges to 0. Therefore, a trained D would have $\theta = 0$ and the sampling would cover the distribution perfectly: $\text{GO}(D_\theta, x_0^{\text{gen}}) = 0$.

Regarding generalization, consider a new 1-D underlying data distribution where the samples lie on a grid $x^{\text{real}} \in X^{\text{real}} \subseteq \{kn | n \in \mathbb{Z}\}$ (for some fixed scalar k). Suppose we have only two real datapoints, $x_0^{\text{real}} = 0$ and $x_1^{\text{real}} = 2\pi$. For the discriminator, we choose $D(x) = \cos(\theta x)$. Assuming a broad uniform distribution P for x_0^{gen} and appropriate T and η, GO will converge to the nearest local maximum of D to x_0^{gen}, i.e. $x^{\text{gen}} \in \{2\pi n/\theta | n \in \mathbb{Z}\}$.

The expected value of the loss function can be calculated as

$$\begin{aligned} E[L_D] &= E[D(x^{\text{gen}})] - 0.5D(x_0^{\text{real}}) - 0.5D(x_1^{\text{real}}) \qquad (9)\\ &= E[1] - 0.5D(0) - 0.5D(2\pi) = 1 - 0.5 - 0.5\text{cos}(\theta 2\pi). \qquad (10) \end{aligned}$$

Taking the derivative of $E[L_D]$ with respect to θ yields $dE[L_D]/d\theta = \pi \text{sin}(\theta 2\pi)$, where gradient descent on θ converges to $\theta = 1$ if appropriate learning rates are used and GO starts sufficiently close.[1] Then, DOG generalizes to the underlying (grid) data distribution, because we can also generate samples that are not among the real samples: For example, $x^{\text{gen}} = \text{GO}(D_1, 4\pi + \epsilon) = 4\pi$ for $x_0^{\text{gen}} = 4\pi + \epsilon$ and a small ϵ.

We can now extend this example to higher dimensions by overlaying independent distributions and discriminators. For 2 dimensions, we have $x^{\text{real}} = (x_1^{\text{real}}, x_2^{\text{real}}) \in X^{\text{real}} \subseteq \{(k_1 n_1, k_2 n_2) | n_1, n_2 \in \mathbb{Z}\}$ with $D(x) = \text{cos}(\theta_1 x_2)\text{cos}(\theta_2 x_2)$. Convergence and generalization follow the same pattern as in the 1-D case.

C.3 Score Surface

Consider the score surface defined by D (see Figure 2(a)) to gain another perspective. Assuming a smooth surface and converging GO, the generated samples lie at local maxima of the score surface (local minima of the L_G surface). L_D increases the scores of the real samples, potentially creating new local maxima, while it decreases the scores of the generated samples, potentially destroying their corresponding local maxima. While this process could introduce new local maxima elsewhere, which might then be destroyed again, it is worth considering a situation where there is exactly one strict local maximum at each real sample, and no other local maxima after training. This means that D has memorized every sample in the training set, since by assumption GO always ends at local maxima.

Note that GAN optimization seeks a Nash equilibrium Heusel et al. (2017) between G and D, where neither has an advantage by changing its weights. Suppose further that all local maxima are sampled with equal probability for DOG, i.e., the real and generated samples are drawn from the same distribution of real samples R. In this situation, we would have reached such a Nash equilibrium between GO and D: GO has no learnable parameters and therefore cannot reduce any loss, and for D_θ the terms of L_D cancel out and there is no incentive for D to change its weights. In this game, only one player, D, can choose its action (by changing its weights) and the other player, GO, is fixed given D. For example, for Wasserstein losses, we have in expectation

$$L_D = E_{x_0^{\text{gen}} \sim P}[D_\theta(\text{sg}(\text{GO}(D_\theta, x_0^{\text{gen}})))] - E_{x^{\text{real}} \sim R}[D_\theta(x^{\text{real}})] \tag{11}$$

$$= E_{x^{\text{gen}} \sim R}[D_\theta(x^{\text{gen}})] - E_{x^{\text{real}} \sim R}[D_\theta(x^{\text{real}})] = 0 \tag{12}$$

and hence $\nabla_\theta L_D = 0$.

However, in practice, pure memorization of the training samples is not desirable, as we want the distribution of the generated samples from $\text{GO}(D, P)$ to match the underlying distribution of the real samples. We are interested in generating new samples from the domain and thus in generalization. For GANs, generalization comes from an imperfect discriminator or an imperfect generator Zhang et al. (2018); Brock et al. (2019). Similarly, generalization in DOG must originate from an imperfect D that also has other local maxima at other realistic samples from the domain or from an imperfect GO that does not converge to local maxima but still ends at a realistic sample.

C.4 Density

We can obtain the density $p(x^{\text{gen}})$ of a generated sample x^{gen} by counting the GO paths that end there and weighting them by the density of x_0^{gen}:

$$p(x^{\text{gen}}) \propto E_{x_0^{\text{gen}} \sim N(0,I)}[I(\text{GO}(D_\theta, x_0^{\text{gen}}), x^{\text{gen}})] \tag{13}$$

Here, $I(a, b)$ is the indicator function that returns 1 if $a = b$ and 0 otherwise.

The set of local maxima of the discriminator D can be defined as $\text{locmax}(D) := \{x | x \text{ is a local maximum of } D(x)\}$. Assuming that the gradient descent GO always converges to a local maximum $x \in \text{locmax}(D)$, we can partition the space of starting locations according to the

[1]Note that other values, such as $\theta = 2$ would also minimize L_D, which can be reached by starting the training close to these values. The choice of the discriminator as well as the initialization leads to different generalization patterns.

local maxima that are reached, turning the density p into a probability if the set is discrete (i.e., if all local maxima are strict).

If we further assume that D is smooth, all local maxima are strict, and η is sufficiently small, then there exists a small radius r around each $x \in \text{locmax}(D)$ such that for every x_0^{gen} in this radius ($||x_0^{\text{gen}} - x||_2 < r$), we have $I(\text{GO}(D, x_0^{\text{gen}}), x) = 1$, i.e. the GO paths lead to the closest local maximum. This implies that $\forall x \in \text{locmax}(D) : p(x) > 0$, since the starting distribution $P = N(0, I)$ is positive everywhere, i.e., every local maximum will eventually be sampled. This affects both training and inference: Only those local maxima that actually receive samples x^{gen} can be used by the loss function L_D during training and are relevant for inference.

Note, however, that $p(x)$ may be higher for some x than for others, meaning that more gradient descent paths (weighted by their starting density $P(x_0^{\text{gen}})$) end at some x than at others. Assuming good performance, where all samples are realistic and $p(x)$ largely follows the real data distribution, the paths that emerge are meaningful. The emergence of these paths is non-trivial: it is not enough to have realistic samples at local maxima; they must also be reachable via the GO paths with an appropriate probability. In particular, the loss function L_D only applies directly at the real and generated samples, not at distant locations (i.e., x_0^{gen}). However, the gradient at these locations may already determine the general direction of many GO paths. Investigating why and how exactly these paths emerge to follow the underlying data distribution using only L_D is an interesting direction for future work.

C.5 Assumptions

We will now assess the validity of the previously made assumptions.

Firstly, the assumption of smoothness holds for all discriminators that utilize differentiable activation functions. Furthermore, for discriminators that are almost everywhere differentiable, such as those using ReLUs, it is unlikely to encounter non-differentiable points due to numerical reasons.

Regarding the assumption of GO convergence, as long as η is appropriately small and T is sufficiently large, the GO process will come arbitrarily close to a local maximum in the score surface. In practice, however, we often empirically select η and T for good generation performance without necessarily achieving full convergence.

Finally, we assume the absence of non-strict local maxima, which can be avoided by training discriminators with a gradient penalty Gulrajani et al. (2017). With this technique, the gradients provided by the discriminator on its inputs maintain specific values almost everywhere, thus preventing the existence of flat regions required for non-strict local maxima.

While these assumptions are useful for theoretical analyses, they may not always hold true in practice. Nonetheless, as discussed in Appendix C.3, an imperfect GO for example might even be a requirement for generalization.

C.6 Training Dynamics

Note that for successful DOG training runs, we are usually able to obtain samples with realness scores close to random guessing. For example, in Figure 2(c) and Table 2, the Wasserstein L_D is close to zero but not substantially below it, indicating that the generated and real samples receive similar scores. If the generated samples were given low scores by the discriminator, the training would collapse because the discriminator would only be trained to assign low scores to samples it had already assigned low scores to before. However, if GO can find holes in the loss surface by achieving higher realness scores than real samples, there is an incentive for the discriminator to fill the holes by ignoring the input or by outputting a score close to random guessing.

These considerations further motivate the use of gradient penalty Gulrajani et al. (2017) or spectral norms Miyato et al. (2018) in the discriminator to avoid having zero gradients on its input for GO. In practice, we also observe that the loss of the DOG discriminator varies less compared to GAN training, which is potentially relevant for tuning the weights of regularization terms such as bCR.

D COMPUTATIONAL COST

To obtain an approximate comparison for the computational cost of DOG and GANs, we assume that the cost of each optimizer (i.e., weight/sample and momentum updates) is negligible. We also assume that the cost of a forward pass through D is equal to the cost of a forward pass through a corresponding GAN generator model G, as G and D are often designed to be similar in size Karras et al. (2020). Additionally, we assume that a backward pass is twice as expensive as a forward pass.

For inference, a DOG GO requires T forward and backward passes through D. Therefore, a single DOG generation costs $3T$ times as much as a generation with a G that requires only a single forward pass through G. This brings the DOG inference cost closer to the cost of denoising diffusion models that perform dozens to thousands of forward passes.

When it comes to training, assuming a vanilla GAN with no augmentations or regularizations, we have a G update step with one forward and backward pass through G and D (cost 6), and a D update step with one forward pass through G and two forward and backward passes through D, one for x^{gen} and one for x^{real} (cost 7). DOG does not perform a generator update step like GANs since there is no G. The discriminator update step includes a GO to obtain x^{gen} ($3T$ passes) and again two forward and backward passes through D (cost 6). In total, we have 13 passes for a GAN training step versus $3T + 6$ passes for a DOG training step. For a typical $T = 100$, this theoretically makes DOG ≈ 23.5 times slower than GANs.

In practice, we may need to use larger discriminators for DOG, since they have to solve different tasks. However, we could also achieve higher accelerator utilization since no data loading bottlenecks can occur in the DOG GO. Additionally, as quantified in the hyperparameter study (Section 3.3), overall fewer training steps (and less time) might be needed for DOG, because there is no generator whose updates lag behind D, providing high quality x^{gen} for training D directly. While a longer overall training time is acceptable for domains that are currently not computationally constrained (such as graphs with GANs), this becomes problematic for the image domain. Finding ways to reduce T should be the focus of future work.