Toward a Generalization Metric for Deep Generative Models

Hoang Thanh-Tung
Deakin University
hoangtha@deakin.edu.au

Truyen Tran
Deakin University
truyen.tran@deakin.edu.au

Abstract

Measuring the generalization capacity of Deep Generative Models (DGMs) is difficult because of the curse of dimensionality. Evaluation metrics for DGMs such as Inception Score, Fréchet Inception Distance, Precision-Recall, and Neural Net Divergence try to estimate the distance between the generated distribution and the target distribution using a polynomial number of samples. These metrics are the target of researchers when designing new models. Despite the claims, it is still unclear how well can they measure the generalization capacity of a generative model. In this paper, we investigate the capacity of these metrics in measuring the generalization capacity. We introduce a framework for comparing the robustness of evaluation metrics. We show that better scores in these metrics do not imply better generalization. They can be fooled easily by a generator that memorizes a small subset of the training set. We propose a fix to the NND metric to make it more robust to noise in the generated data. Toward building a robust metric for generalization, we propose to apply the Minimum Description Length principle to the problem of evaluating DGMs. We develop an efficient method for estimating the complexity of Generative Latent Variable Models (GLVMs). Experimental results show that our metric can effectively detect training set memorization and distinguish GLVMs of different generalization capacities.

1 Introduction

Deep Generative Models [e.g. 7, 8, 10, 18, 32] use deep neural networks to model the target distributions. Evaluating DGMs is hard because (1) the distributions of interest are often high dimensional, (2) the likelihood functions are not always available or easily computable. A common way to evaluate a DGM is to measure how close the generated distribution p_g is to the target distribution p_r. Because the sample complexity of traditional metrics such as KL divergence or Wasserstein distance is exponential in the dimensionality of the distribution, they cannot be used for real world distributions. New evaluation metrics with polynomial sample complexity such as Fréchet Inception Distance (FID) [14], Inception Score (IS) [27], and Precision-Recall [20, 26, 28] are commonly used and are the target of many researchers when designing new models. The reduced sample complexity comes at the cost of reduced discriminative power. These metrics cannot tell the difference between a model that memorizes the training data and a model that generalizes. Gulrajani et al. [13] applied the Neural Net Divergence (NND) [2] to the problem and claimed that NND can measure the generalizability of generative models (GMs).

In this paper, we investigate the capacity of the above metrics in measuring the generalization capacity of DGMs. We perform worst-case analyses for these metrics and found that all of them cannot measure generalization, i.e. better scores in these metrics do not imply better generalization capacities. A generator that memorizes a small subset of the training data can achieve (near) optimal scores in these metrics. Our results suggest that using samples alone is not enough to estimate the generalization capacities of DGMs.

1st I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020), Vancouver, Canada.

We propose a fix to the NND to make it robust to noise in the generated data. Our fixed NND can be used to estimate the generalization capacity of GMs.

Notations

p_r: the unknown target (real) distribution in $\mathbb{R}^{d_x}$. p_g: the model generated (fake) distribution in $\mathbb{R}^{d_x}$. p_z: the latent distribution in $\mathbb{R}^{d_z}$. $\boldsymbol{z} \sim p_z$: a latent variable. $\boldsymbol{x} \in \mathbb{R}^{d_x}$: a data point. $G(\cdot;\boldsymbol{\theta})$: a generative model with parameter $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$. $\mathcal{D}_r = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_k \mid \boldsymbol{x}_i \sim p_r\}$: a set of samples from p_r. $\mathcal{D}_g = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_l \mid \boldsymbol{x}_i \sim p_g\}$: a set of samples from p_g. $\mathcal{D}_{train} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n \mid \boldsymbol{x}_i \sim p_r\}$: the training set, $|\mathcal{D}_{train}| = \text{poly}(d_x)$. $\mathcal{D}_{test} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m \mid \boldsymbol{x}_i \sim p_r,\ \boldsymbol{x}_i \notin \mathcal{D}_{train}\}$: the test set, $|\mathcal{D}_{test}| = \text{poly}(d_x)$. $\hat{p}_{\mathcal{D}}$: the uniform distribution over $\mathcal{D}$.

2 Background and Related Work

2.1 Divergence based evaluation metrics

A generative model G is trained on $\mathcal{D}_{train}$ to produce p_g that hopefully approximates p_r. The quality of G is defined by the distance/divergence between p_r and p_g. Let f be a distance/divergence function. Because $f(p_r, p_g)$ is intractable for most distributions of interest, we approximate it with $f(\hat{p}_r, \hat{p}_g)$ where $\mathcal{D}_r, \mathcal{D}_g$ polynomial sized datasets.

Inception Score (IS) [27] is defined as: $\text{IS}(p_g) = \exp\left(\text{KL}\left(p(y|\boldsymbol{x})||p(y)\right)\right)$ where $\boldsymbol{x} \sim p_g$, $p(y) = \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[p(y|\boldsymbol{x})\right]$. $p(y|\boldsymbol{x})$ is computed by feeding a generated datapoint through a pretrained classifier, e.g. the Inception net [29]. IS is maximized when $\text{H}(y|\boldsymbol{x}) = 0$ and $\text{H}(y) = \ln C$ where C is the number of classes. Barratt and Sharma [4] showed that $\text{IS}(\hat{p}_g) \leq C$. To make our writing consistent, we convert all of the metrics in this paper to (pseudo) divergences. A lower divergence should imply that p_g is closer to p_r. We define the following pseudo divergence:

$$f_{is}(p_g) = C - \exp\left(\text{KL}\left(p(y|\boldsymbol{x})||p(y)\right)\right) \tag{1}$$

This is a pseudo divergence because (1) it only contains 1 distribution, p_g, the other distribution is implicitly the dataset on which the classifier was trained, (2) $f_{is}(p_g) = 0$ does not imply that the two distributions are the same (more on this in Sec. 3). A *fundamental flaw* of IS is that a smaller $f_{is}(p_g)$ implies that $p(y)$ is closer to the uniform distribution, not that $p(y)$ is closer to the true distribution over the labels. Fortunately (or unfortunately), the distribution over the labels in Imagenet dataset [6] is uniform. This makes IS a popular and somewhat abused tool for evaluating image generation models.

Fréchet Inception Distance (FID): Heusel et al. [14] approximate p_r, p_g with $\mathcal{N}(\boldsymbol{\mu}_r, \Sigma_r)$ and $\mathcal{N}(\boldsymbol{\mu}_g, \Sigma_g)$, respectively, and then compute the Fréchet distance between the 2 Gaussians:

$$f_{fid}(p_r, p_g) = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right) \tag{2}$$

Because FID compares only the first 2 moments of p_r and p_g, $f_{fid}(p_r, p_g) = 0$ does not guarantee that the 2 distributions are the same.

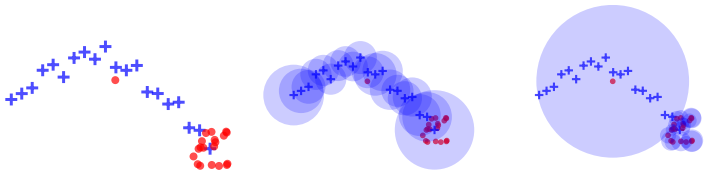
Neural Net Divergence (NND) [2, 13] uses neural nets to compute the divergence between p_r and p_g

$$f_{nnd}(p_r, p_g) = \sup_{f_{\boldsymbol{\theta}} \in \mathcal{F}} \left(\mathbb{E}_{\boldsymbol{x}\sim p_r}\left[f_{\boldsymbol{\theta}}(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[f_{\boldsymbol{\theta}}(\boldsymbol{x})\right]\right) \tag{3}$$

where $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_x} \to \mathbb{R}$ is usually a neural network with finite capacity. $f_{\boldsymbol{\theta}}$ is trained to maximize the difference in Eqn. 3. Arora et al. [2] showed that NND can detect the lack of diversity in generated data and Gulrajani et al. [13] it used to estimate the generalizability of GMs. We show in Sec. 3 that the diversity in generated data can easily be faked by adding white noise to the data. This fake diversity is present in real world models such as GANs and VAEs. We propose a fix to NND that removes the effect of fake diversity while retaining the discriminative power of NND.

2.2 Precision-Recall based evaluation metrics

The above metrics use scalars to measure both the quality and diversity of generated samples. The following metrics are designed to separate quality from diversity.



(a) (b) (c)

Figure 1: k-nn PR on a 2-dimensional dataset. (a) real data (blue crosses) and fake data (red dots). (b), (c) real and fake manifolds approximated with 3-nearest neighbors. Although the real and the fake data are very different, the precision and recall of the fake data are 1 and 1.

k-means based Precision-Recall (k-means PR) [26] applies k-means algorithm to $\mathcal{D}_r \cup \mathcal{D}_g$, builds 2 histograms to approximate p_r and p_g, and computes the difference between these histograms. The results are 2 F-scores, F_β and $F_{1/\beta}$. For $\beta > 1$, F_β can be interpreted as the recall (diversity) and $F_{1/\beta}$ can be interpreted as the precision (quality) of the generated data (see Algo. 1 in Appx. C). We define the following pseudo divergences: $f_{p\text{-}kmeans} = 1 - F_{1/\beta}$, $f_{r\text{-}kmeans} = 1 - F_\beta$.

k-nn based Precision-Recall (k-nn PR) [20] computes the precision and recall by comparing the reconstructed real manifold and the reconstructed fake manifold (see Algo. 2 in Appx. D and Fig. 1). Kynkäänniemi et al. [20] claimed that k-nn PR has better discriminative power than k-means PR and demonstrated that on several datasets. However, we show in Sec. 3 that k-nn PR has the worst worst-case performance and in practice it cannot differentiate a good model from a very bad model. We define the following pseudo divergences: $f_{p\text{-}knn} = 1 - P$, $f_{r\text{-}knn} = 1 - R$.

Similar to IS and FID, k-means PR and k-nn PR assign (near) perfect scores to a generative model that produce the entire training set.

2.3 Other evaluation metrics

Metrics for class-conditional models: Esteban et al. [9] proposed the "Train on Synthetic Test on Real" (TSTR) approach to evaluate class-conditional GANs. The metric is expensive to compute as it requires training a classifier on a labeled synthetic dataset. Shmelkov et al. [28] extended the idea to a Precision-Recall like metric for class-conditional GANs. Because these metrics are applicable only to class-conditional GMs, we do not consider the metric in our paper.

Topological/Geometrical approaches: Horak et al. [15], Khrulkov and Oseledets [17] applied geometrical and topological tools to measure the difference between the fake manifold and the real manifold. These metrics are not designed for measuring generalizability and are note considered in this paper. Karras et al. [16] used the *pairwise perceptual distance* - the pairwise distance between feature vectors on the feature manifold - to measure the smoothness of the feature manifold. The intuition is that if the perceptual distance is small then the manifold is smoother and the GM might learn better representations and have better generalizability. We show in Appx. E that the pairwise distance does not well reflect the smoothness of the manifold and it is minimized when total mode collapse occurs. Therefore, pairwise perceptual distance is not a good metric for GMs.

Non-parametric approaches: Arora et al. [3] used the birthday paradox and human evaluation to estimate the number of distinct modes that a GM can generate. The method cannot be automated and has high variance and bias because humans are involved in the loop. Meehan et al. [21] proposed a three sample test for data copying in GMs. The metric can detect data copying but cannot measure the generalizability of a model or the quality of the generated samples.

2.4 A generative model with poor generalization capacity

When mode collapse (training set memorization) [10] happens, the generative model generates slightly different versions of the memorized datapoints. We approximate this kind of behavior with the following generative procedure:

Given a dataset $\mathcal{D} \subseteq \mathcal{D}_{train}$, a constant $\epsilon \in \mathbb{R}$, a new datapoint is generated by

1. Draw a datapoint $\boldsymbol{x} \in \mathcal{D}$ at random

2. Draw a noise vector $\boldsymbol{u} \sim p_u$, where p_u is the noise distribution.
3. Output the noisy datapoint: $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \boldsymbol{u}$

In our experiments, $p_u = \mathcal{U}(-1, 1)$ (see Fig. 5 in Appx. F for samples from MNIST dataset). We tried replacing $\mathcal{U}(-1, 1)$ with $\mathcal{N}(\mathbf{0}, I)$ and found no difference in performance. This generator, denoted as $G_{\mathcal{D},\epsilon}$ has no real generalization capacity and the number of essentially different modes in $p_{G_{\mathcal{D},\epsilon}}$ is $|\mathcal{D}|$. For $\epsilon > 0$, this generator can be approximated by a neural net with finite capacity.

3 Evaluating evaluation metrics

Given a training set $\mathcal{D}_{train} \sim p_r$, we train a generative model G on $\mathcal{D}_{train}$ and get a distribution p_g. We would like to estimate the distance/divergence between p_r and p_g using a divergence/distance function f. Because computing $f(p_r, p_g)$ is intractable, we estimate $f(p_r, p_g)$ using $f(\mathcal{D}_{test}, \mathcal{D}_g)$ where $\mathcal{D}_{test} \sim p_r$ and $\mathcal{D}_g \sim p_g$. If $\mathcal{D}_{test} = \mathcal{D}_{train}$ then G can achieve the perfect score by memorizing the entire $\mathcal{D}_{train}$. To reduce the effect of training set memorization, we require that $\mathcal{D}_{test} \cap \mathcal{D}_{train} = \emptyset$. We want a divergence f that is robust against training set memorization. Specifically, we want f to satisfy:

$$\mathbb{E}_{\mathcal{D}_{test} \sim p_r,\ |\mathcal{D}_{test}|=m} \left[f(\mathcal{D}_{test}, \mathcal{D}) \right] > \mathbb{E}_{\mathcal{D}_{test} \sim p_r,\ |\mathcal{D}_{test}|=m} \left[f(\mathcal{D}_{test}, \mathcal{D}_{train}) \right] \quad \text{for all } \mathcal{D} \subset \mathcal{D}_{train},\ |\mathcal{D}| < |\mathcal{D}_{train}| \tag{4}$$

We show in the next subsections that Inception Score, Precision-Recall, and Neural Net Divergence are not robust against training set memorization.

3.1 Worst-case analyses of evaluation metrics

Let $\mathcal{D}^*$ be the smallest subset of $\mathcal{D}_{train}$ that satisfy $\mathbb{E}_{\mathcal{D}_{test} \sim p_r,\ |\mathcal{D}_{test}|=m} \left[f(\mathcal{D}_{test}, \mathcal{D}) \right] \leq \mathbb{E}_{\mathcal{D}_{test} \sim p_r,\ |\mathcal{D}_{test}|=m} \left[f(\mathcal{D}_{test}, \mathcal{D}_{train}) \right]$. The divergence f cannot distinguish a model that produces only $\mathcal{D}^*$ and a model that produce $\mathcal{D}_{train}$. By memorizing only $\mathcal{D}^*$, a generator can fool f that it can produce a larger dataset. The smaller $|\mathcal{D}^*|$ is, the more vulnerable f is to training set memorization. We construct $\mathcal{D}^*$ for the metrics mentioned above and discuss the relationship between $\mathcal{D}^*$ and the performance of a metrics in practice. Because NND is the only metric that is designed to be robust against training set memorization, it is the focus of this section. We defer the details for other metrics to the appendix.

Inception Score

$$\mathcal{D}^*_{is} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_C \mid \boldsymbol{x}_i \in \mathcal{C}_i\} \tag{5}$$

where $\mathcal{C}_i \subset \mathcal{D}_{train}$ is the i-th class of the training set. Details in Appx. A.

k-means Precision-Recall

$$\mathcal{D}^*_{kmeans} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_k \mid \boldsymbol{x}_i \in \mathcal{C}_i\} \tag{6}$$

where $\mathcal{C}_i$ is the i-th cluster of the k-means clustering algorithm. Details in Appx. C.

k-nn Precision-Recall

$$\mathcal{D}^*_{knn} = \{\boldsymbol{x}_1,\ \boldsymbol{x}_2 \mid \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}_{train} \text{ and } \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \geq \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \, \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_{train}\} \tag{7}$$

Details in Appx. D.

Neural Net Divergence is computed by training a finite capacity neural net $f_{\boldsymbol{\theta}}$ to maximize the objective in Eqn. 3. Because the value of NND is dependent of the network and the training procedure, there is no fixed $\mathcal{D}^*_{nnd}$. However, given a network, we can still estimate $\mathcal{D}^*_{nnd}$ through experiments.

First, we investigate the effect of noise on NND. In our experiment, the dataset is the MNIST dataset, $f_{\boldsymbol{\theta}}$ is a MLP with 3 hidden layers and 512 neurons in each layer. Details about the architecture and hyper parameters are in Appx. B. We use the generator described in Sec. 2.4 to generate noisy versions of datapoints in $\mathcal{D}$, a random subset of $\mathcal{D}_{train}$ (see Fig. 5 in Appx. F for noisy images generated by the generator). The number of essentially different datapoints that the generator can

generate is $|\mathcal{D}|$. We vary $|\mathcal{D}|$ from 1000 to 60000 and ϵ from 0 to 1. The noisy samples are generated continuously throughout the training process as recommended in [13]. As we can see in Fig. 5, for $\epsilon = 0.1$ and $\epsilon = 0.5$ the noisy images are very clear and easily recognizable. For $\epsilon = 1$, the images are much more noisy but still recognizable. At the first glance, one might think that adding noise to clean data will increase the NND because it worsen the quality of the data. The result in Fig. 2(a) shows the opposite. As we can see from the figure, when ϵ and $|\mathcal{D}|$ increase, f_{nnd} decreases. Increasing ϵ results in a bigger decrease in f_{nnd} than increasing $|\mathcal{D}|$. For $\epsilon = 1$, a noisy random subset of size 1000 outperforms a noiseless set of size 60000 by a large margin. The network also cannot distinguish noisy sets of size 10000 and 60000 and gives them the same score (the red solid line in Fig 2(a)). We conclude that $|\mathcal{D}^*_{nnd}| \leq 1000$ for this specific network architecture.[1]

On the positive side, we note that when $\epsilon = 0$, the NND decreases as $|\mathcal{D}|$ increases. If we can remove the fake diversity caused by noise, then NND can be used to estimate the diversity of generated data. We note that the NND is plateauing as $|\mathcal{D}|$ reaches 60000. If we continue to increase $|\mathcal{D}|$ to much larger numbers, the network will assign about the same NND to these datasets. This behavior is expected as the network has a finite capacity (polynomial in d_x). In order to use NND to test for generalization, we need to use a network that can distinguish datasets of size at least $|\mathcal{D}_{train}|$.

The above experiment shows that diversity can be faked by adding random noise to the data. We can reduce the effect of noise by generating a noisy data set $\mathcal{D}_g$ of a fixed (and polynomial) size and train $f_{\boldsymbol{\theta}}$ on $\mathcal{D}_g$ and $\mathcal{D}_{test}$. We use $G_{\mathcal{D},\epsilon}$ to generate a dataset $\mathcal{D}_g$ of 60000 noisy datapoints. The result is shown in Fig. 2(b). In contrast to the previous experiment, the NND increases with the level of noise. This behavior is expected as a larger ϵ results in worse images, making the task of separating $\mathcal{D}_g$ from $\mathcal{D}_{test}$ easier. By fixing the size of $\mathcal{D}_g$, we can effectively remove the fake diversity.

The outputs of GMs usually contain noise. For autoregressive models and energy based models, the noise comes from the randomness in the sampling process. For generative latent variable models like GANs, VAEs, and normalizing flows, the noise comes from the input distribution. A Generative Latent Variable Model (GLVM) can memorize a subset of the training data, add the input noise to the memorized datapoints, and fool f_{nnd} that it has learn to perfectly reproduce the target distribution. Our next experiment investigate this phenomenon in real world models. We use GANs as our example. We select a random subset $\mathcal{D}$ from $\mathcal{D}_{train}$, use that for training a GAN and use the resulting model to generate fake data. For each model, we compute 2 NND scores following original procedure and our new procedure described in the previous paragraph. Fig. 2(c) shows the changes in NND of models over 500 epochs. As expected, the NND of 'Fix' models is higher than that of 'Inf' models. However, the difference between 'Fix' models and 'Inf' models is small, suggesting that the level of noise in the outputs of GAN is much lower than that in our experiment in Fig. 2(a).

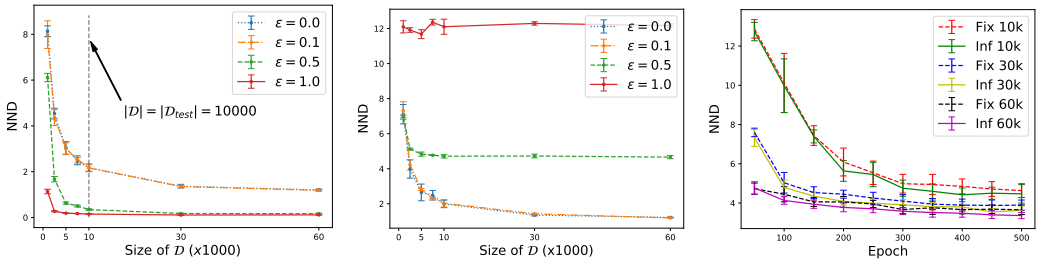
3.2 Comparing evaluation metrics

We measure the performance of evaluation metrics on GANs trained on MNIST to see how their worst case performance correlate the their real world performance. The result is shown in Fig. 3(a) - 3(c).[2] We trained 5 variants of GAN on the MNIST dataset. The variants are GAN with 0GP regularizer [31] (GAN0GP), GAN with R1 regularizer [22] (GANR1), WGAN with gradient penalty [1, 12] (WGAN1GP), WGAN with 0GP [31], and WGAN1GP with our new constant speed regularizer (WGAN1GP const) (introduced in Appx. E). Thanh-Tung et al. [31] showed in their paper that 0GP improves the generalization capacity of the discriminator and that in turn, improves the generalization capacity of the generator. They also claimed that 1GP [12] does not help improving generalization. We verify their claims with our fixed NND metric.

The result for NND is shown in Fig. 3(c). WGAN0GP consistently has lower NND than WGAN1GP. That confirms the claims by Thanh-Tung et al. [31]. We also find that GAN0GP also more stable than GANR1 as GANR1 can suddenly collapse during training. Fig. 3(c) shows that GANR1 suffers from overfitting after epoch 400 and its NND starts to increase. GAN0GP's NND decreases as the training progresses. Our constant speed does a very good job at improving the generalization capacity of WGAN1GP. WGAN1GP const has almost the same NND as WGAN0GP while producing smoother interpolation (see Fig. 3(e), 3(f)).

[1] We note that our experiments with $p_u = \mathcal{N}(0, I)$ produces the same results.

[2] IS and FID are not included because computing them requires the Inception model. Gulrajani et al. [13] empirically showed that IS and FID are not robust against training set memorization. We refer readers to their experimental results.



(a) (b) (c)

Figure 2: The effect of noise on NND. (a): noisy samples were generated continuously throughout training. (b): a fixed number of noisy samples were used in training. (c): continuously generated fake data (labeled 'Inf + training set size') v.s. fixed size fake data (labeled 'Fix + training set size').

k-means PR does not do a good job at distinguishing models with different generalization capacities. Fig. 3(a) shows that all models achieve almost perfect precision (i.e. $F_{1/\beta}$) after 400 epochs. In contrast to NND, k-means PR assigns the highest precision and recall to GANR1. This result is expected as Thanh-Tung and Tran [30] and Thanh-Tung et al. [31] suggested that R1 regularizer may encourage the generator to memorize the training set. The other GANs with better generalization capacity will produce more datapoints that are further away from the training and test sets. The far datapoints result in lower recall as we see in Fig. 3(a).

k-nn PR does a much better job at distinguishing different models (Fig. 3(b)). The result seems to justify the claim of its author [20]. However, in Fig. 3(d) shows that k-nn PR assigns very high recall to a very bad GAN. An illustration of this phenomenon is given in Fig. 1. k-nn works fine for good models that distribute the fake data evenly across the real manifold but it is vulnerable to bad models that generate many outliers.

Discussion: This experiment shows that the robustness of evaluation metrics on real world datasets is directly correlated to the size of their minimal dataset $\mathcal{D}^*$. k-nn PR has the smallest $|\mathcal{D}^*|$ and is the most vulnerable to outliers in the data. k-means PR has $|\mathcal{D}^*_{kmeans}| = k$ and different datapoints in $\mathcal{D}^*_{kmeans}$ must lie in different clusters. To fool k-means PR, a generator must produce fake datapoints belonging to different clusters. That is nontrivial for real world generative models. NND (both the original and our updated version) has the biggest $\mathcal{D}^*_{nnd}$ and is the most robust. Our updated NND can be used to estimate the generalization capacity of GMs.

4 Future direction

The Minimum Description Length (MDL) principle is a framework for comparing the generalization capacities of different models. The main challenge in apply MDL to deep models is that it is very hard to measure the complexity of a neural network. In Appx. E, we present some initial experiments of our new algorithm for estimating the complexity of Generative Latent Variable Models.

5 Conclusion

In this paper, we investigate the ability to measure generalization of different evaluation metrics. We show that all of the existing metrics can be fooled by training set memorization. We study the worst-case performance of common metrics and show that their worst-case performance is related to their robustness on real world scenarios. We identify the weakness of NND and propose a fix to the problem. Our experiments show that the fixed NND metric is robust to noise in the input and can be used to estimate the generalization capacities of generative models.

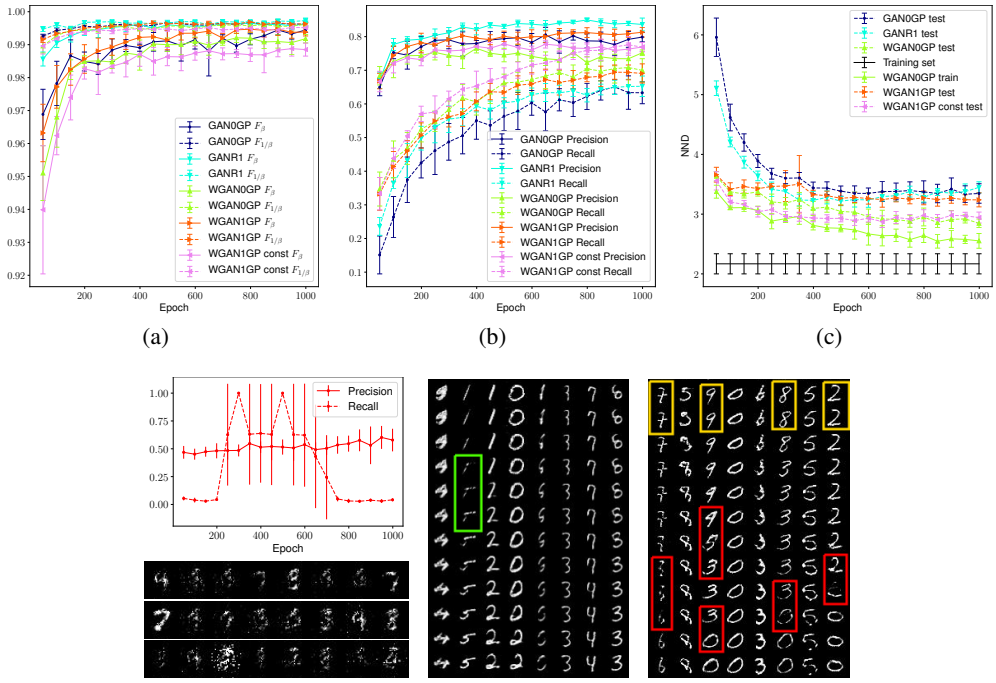


Figure 3: We trained 2 sets of GANs with 2 different learning rate. We deliberately chose a large learning rate for the second group to make it collapse. (a) - (c) results of k-means PR, k-nn PR, and NND of models trained with $lr_1 = 2 \times 10^{-4}$. (c) Because $f_{nnd}(\hat{p}_{train}, \hat{p}_g)$ and $f_{nnd}(\hat{p}_{test}, \hat{p}_g)$ always show the same trend, we show only $f_{nnd}(\hat{p}_{test}, \hat{p}_g)$. We show $f_{nnd}(\hat{p}_{train}, \hat{p}_g)$ for WGAN0GP as a representative. (d) top: k-nn PR of GAN0GP trained with $lr_2 = 10^{-3}$, bottom: generated samples from a GAN with recall equal 1. (e), (f) interpolation result of WGAN1GP-const and WGAN1GP at epoch 1000. Despite the lower NND, WGAN0GP produces similar interpolation result as WGAN1GP.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[2] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[3] Sanjeev Arora, Andrej Risteski, and Yi Zhang. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.

[4] Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

[5] A. Barron, J. Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2016.

[8] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox,

and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3608–3618. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8619-implicit-generation-and-modeling-with-energy-based-models.pdf`.

[9] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[10] 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. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[11] Peter Grunwald. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*, 2004.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

[13] Ishaan Gulrajani, Colin Raffel, and Luke Metz. Towards GAN benchmarks which require generalization. In *International Conference on Learning Representations*, 2019.

[14] 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. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc., 2017.

[15] Danijela Horak, Simiao Yu, and Gholamreza Salimi-Khorshidi. Topology distance: A topology-based approach for evaluating generative adversarial networks. *arXiv preprint arXiv:2002.12054*, 2020.

[16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[17] Valentin Khrulkov and Ivan Oseledets. Geometry score: A method for comparing generative adversarial networks. *arXiv preprint arXiv:1802.02664*, 2018.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[19] Andrei Nikolaevich Kolmogorov. Three approaches to the quantitative definition of information. *International journal of computer mathematics*, 2(1-4):157–168, 1968.

[20] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, pages 3929–3938, 2019.

[21] Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. A Non-Parametric Test to Detect Data-Copying in Generative Models. *arXiv e-prints*, art. arXiv:2004.05675, April 2020.

[22] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3478–3487, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[23] J. Rissanen. Paper: Modeling by shortest data description. *Automatica*, 14(5):465–471, September 1978. ISSN 0005-1098. doi: 10.1016/0005-1098(78)90005-5.

[24] Jorma Rissanen. Stochastic complexity and modeling. *The annals of statistics*, pages 1080–1100, 1986.

[25] Jorma Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society: Series B (Methodological)*, 49(3):223–239, 1987.

[26] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, pages 5228–5237, 2018.

Table 1: Configuration for experiment in Fig. 2(a)

Network architecture	MLP with 3 hidden layers, 512 hidden neurons
Loss function	WGAN1GP loss function [12]
Number of training iteration	20000
Learning rate	1e-4
Optimizer	Adam with $\beta_1 = 0.9,\ \beta_2 = 0.999$
$\lvert\mathcal{D}_{test}\rvert$	10000

[27] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[28] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2018.

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

[30] Hoang Thanh-Tung and Truyen Tran. Catastrophic Forgetting and Mode Collapse in Generative Adversarial Networks. In *Proceedings of the International Joint Conference on Neural Networks*, Jul 2020.

[31] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *International Conference on Learning Representations*, 2019.

[32] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1747–1756. JMLR.org, 2016.

A Inception Score

We assume that the Inception model is a perfect classifier, i.e. given a real datapoint $\boldsymbol{x}$ the conditional distribution $p(y|\boldsymbol{x})$ is

$$p(y = i|\boldsymbol{x}) = \begin{cases} 1 \text{ if } i \text{ is the correct label of } \boldsymbol{x} \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

We have $\mathrm{H}(y|\boldsymbol{x}) = 0$. We select C real datapoints from C classes in the training set so $\mathrm{H}(y) = \ln C$. The Inception score for the dataset in Eqn. 5 is C.

B Neural Net Divergence

The configuration for the experiment in Fig. 2(a) and 2(b) is given in Table 1.

The GANs in Fig. 2(c) and 3 also use 3 hidden layer MLPs with the same number of hidden neurons.

C k-means Precision-Recall

Algorithm 1 k-means PR

Inputs: $\mathcal{D}_r,\ \mathcal{D}_g$, hyper parameter β
Outputs: $F_\beta,\ F_{1/\beta}$
$\mathcal{C} \leftarrow kmeans(\mathcal{D}_r \cup \mathcal{D}_g)$
$hist_g[i] \leftarrow \frac{\text{\# fake datapoints in cluster } \mathcal{C}_i}{|\mathcal{D}_g|}$
$hist_r[i] \leftarrow \frac{\text{\# real datapoints in cluster } \mathcal{C}_i}{|\mathcal{D}_r|}$
$curve = pr_curve(hist_r, hist_g)$
$F_\beta = f_score(curve, \beta)$
$F_{1/\beta} = f_score(curve, 1/\beta)$
return $F_\beta,\ F_{1/\beta}$

We describe the procedure for constructing $\mathcal{D}^*_{kmeans}$. Without loss of generality, we assume that $|\mathcal{D}_{test}| = |\mathcal{D}_g|$, and the k-means algorithm always finds the optimal clustering. k-means PR applies k-means clustering algorithm on $\mathcal{D}_{test} \cup \mathcal{D}_g$ to get k clusters. Let $\mathcal{C}_i$ be the i-th cluster, $\mathcal{C}_{i,test}$ be the set of test datapoints in this cluster. From the algorithm above, we see the the perfect precision and recall are achieved when $hist_g = hist_r$. To make $hist_g[i] = hist_r[i]$, the generator G needs to

1. memorize a training datapoint $\boldsymbol{x}$ s.t. $\boldsymbol{x}$ is closest to the center of cluster $\mathcal{C}_i$.
2. duplicate the datapoint $|\mathcal{C}_{i,test}|$ times

The number of distinct datapoints that G has to memorize is k, the number of clusters.

D k-nn Precision-Recall

Algorithm 2 k-nn PR

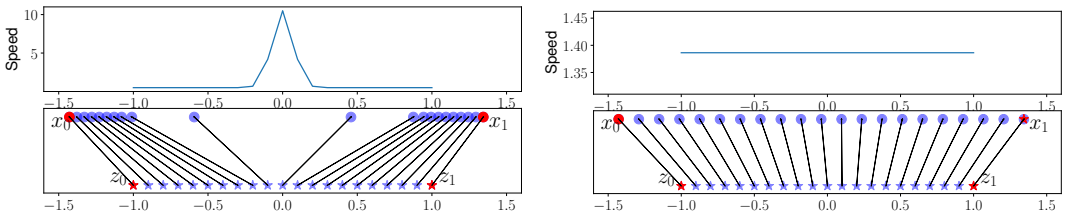
Inputs: $\mathcal{D}_r,\ \mathcal{D}_g$
Outputs: Precision P, Recall R
$\mathcal{M}_g \leftarrow \emptyset$ // build the fake manifold
for each datapoint $\boldsymbol{x}_i \in \mathcal{D}_g$ **do**
  $\boldsymbol{x}_j \leftarrow$ the k-th nearest neighbor of $\boldsymbol{x}_i$
  $S_i \leftarrow sphere(c = \boldsymbol{x}_i, r = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$
  $\mathcal{M}_g \leftarrow \mathcal{M}_g \cup S_i$
end for
build the real manifold $\mathcal{M}_r$
$P \leftarrow \frac{\text{\# fake datapoints in } \mathcal{M}_r}{|\mathcal{D}_g|}$
$R \leftarrow \frac{\text{\# real datapoints in } \mathcal{M}_g}{|\mathcal{D}_r|}$
return $P,\ R$

The idea for constructing $\mathcal{D}^*_{knn}$ is illustrated in Fig. 1. We select $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}_{train}$ s.t. $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \geq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|\ \forall\ \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_{train}$. Because $\boldsymbol{x}_1, \boldsymbol{x}_2$ are in $\mathcal{D}_{train}$, they lie on the real manifold. Thus, the precision is 1. Because $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \geq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|\ \forall\ \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_{train}$, all of the training datapoints are in the two spheres centered at $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. If $\mathcal{D}_{train}$ is large enough then we can assume that all of the test datapoints lies in the manifold reconstructed using $\mathcal{D}_{train}$. Therefore, all of the test datapoints are highly likely to lie in the two spheres centered at $\boldsymbol{x}_1, \boldsymbol{x}_2$ and the recall is 1.

E An MDL inspired Generalization Metric

E.1 Motivation

E.1.1 Preliminaries



(a) (b)

Figure 4: (a) a GM that memorizes the training set. (b) a GM that produces constant speed interpolation.

Path length and speed: A path from $\boldsymbol{z}_0$ to $\boldsymbol{z}_1$ is a continuous function $\boldsymbol{z}(t) : [0, 1] \to \mathbb{R}^{d_z}$ that satisfies $\boldsymbol{z}(0) = \boldsymbol{z}_0,\ \boldsymbol{z}(1) = \boldsymbol{z}_1$. The Frobenius norm of the Jacobian $\frac{\partial \boldsymbol{z}}{\partial t}$ is the speed of the path at time t. The path has constant speed if $\left\|\frac{\partial \boldsymbol{z}}{\partial t}\right\|_F = \text{const}$. A GLVM is a continuous function $\mathcal{G} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$. $\mathcal{G}$ maps a path from $\boldsymbol{z}_0$ to $\boldsymbol{z}_1$ to a path from $\boldsymbol{x}_0 = \mathcal{G}(\boldsymbol{z}_0)$ to $\boldsymbol{x}_1 = \mathcal{G}(\boldsymbol{z}_1)$. The length of the path from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$ is

$$\ell(\boldsymbol{x}_0, \boldsymbol{x}_1) = \int_0^1 \left\|\frac{\partial \boldsymbol{x}}{\partial t}\right\|_F dt = \int_0^1 \left\|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial t}\right\|_F dt \tag{9}$$

Minimum Description Length principle [5, 19, 23, 24, 25] (see [11] for a comprehensive tutorial) is a formalization of Occam's razor principle. Refined MDL [5] defines the stochastic complexity of a dataset $\mathcal{D}$ given a parametric model (a set of parametric hypotheses) $\mathcal{H}$ as

$$\text{SCOMP}(\mathcal{D}|\mathcal{H}) = \text{LEN}(\mathcal{D}|\hat{H}) + \text{COMP}(\mathcal{H}) \tag{10}$$

where $\text{LEN}(\mathcal{D}|H) = -\log p(\mathcal{D}|H)$ is the description length of $\mathcal{D}$ given a point hypothesis $H \in \mathcal{H}$, $\hat{H} \in \mathcal{H}$ is the point hypothesis that maximizes the probability of $\mathcal{D}$, $\text{COMP}(\mathcal{H})$ is the parametric complexity of $\mathcal{H}$. The model with the smallest $\text{SCOMP}(\mathcal{D}|\mathcal{H})$ is the best model, i.e. the model with the best generalization capacity. Let $\mathcal{H}_L$ be the set of L-Lipschitz functions, then the bigger L is, the more 'essentially different' functions $\mathcal{H}_L$ contains (cf. Sec. 2.6.2 in [11]). L and $\text{COMP}(\mathcal{H}_L)$ are positively correlated. In the following, we use L in place of the parametric complexity of $\mathcal{H}_L$.

E.1.2 Detecting memorization and estimating complexity

A metric for generalization must be able to detect memorization. We study the behavior of DGLVMs when they memorize training data. Fig. 4(a) illustrates the situation where a DGLVM $\mathcal{G}_1$ tries to memorize a training set $\mathcal{D}$ (shown by two red dots). For any latent code $\boldsymbol{z}$ (blue stars), $\mathcal{G}_1$ tries to make $\boldsymbol{x} = \mathcal{G}(\boldsymbol{z})$ (blue dots) as close as possible to a training datapoint. Because generated datapoints are concentrated in 2 clusters, when we interpolate from $\boldsymbol{z}_0$ to $\boldsymbol{z}_1$ at constant speed $\left\|\frac{\partial \boldsymbol{z}}{\partial t}\right\|_F$, the speed $\left\|\frac{\partial \boldsymbol{x}}{\partial t}\right\|_F$ blows up in the middle of the path (top pane in Fig. 4(a)). Fig. 4(b) shows another DGLVM $\mathcal{G}_2$ that produces constant speed path from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$. Although the maximum speed in Fig. 4(a) is much higher than that in Fig. 4(b), the paths in the two figures have the same length. The maximum speed is a better feature for detecting memorization than the path length (see Fig. 6).

From Eqn. 9, we see that if $\left\|\frac{\partial \boldsymbol{z}}{\partial t}\right\|_F = \text{const}$, then the maximum speed $s_{max}(\boldsymbol{x}_0, \boldsymbol{x}_1) = \max_{t\in[0,1]}\left(\left\|\frac{\partial \boldsymbol{x}}{\partial t}\right\|_F\right)$ is proportional to the Lipschitz constant of $\mathcal{G}$ on the path. The parametric complexity of $\mathcal{G}$ can be defined as the expectation of s_{max} over the latent space

$$\text{COMP}(\mathcal{G}) = \mathbb{E}_{\boldsymbol{z}_i, \boldsymbol{z}_j \sim p_z}[s_{max}(\mathcal{G}(\boldsymbol{z}_i), \mathcal{G}(\boldsymbol{z}_j))] \tag{11}$$

We use the average of s_{max} instead of the absolute maximum speed because 1) the absolute max speed has too high variance, making it an uninformative quantity, and 2) the average tracks the cost of describing the generated manifold more closely than the absolute maximum speed.

E.2 Definition

Because $-\log(\mathcal{D}|H)$ is not easily computable in some DGLVMs like GANs, we use the divergence between $\hat{p}_r$ and $\hat{p}_g$ in place of the description length of the training data given the DGLVM. The divergence can be computed using one of the methods in Sec. 2.1.

Definition 1 *Let $\mathcal{D}_{train}$ be a set of i.i.d. samples from a distribution p_r. The 'generalization metric' for a DGLVM $\mathcal{G}$ trained on $\mathcal{D}_{train}$ is defined as:*

$$f_{gen}(\mathcal{G}) = \alpha f(\mathcal{D}_{train}, \mathcal{D}_g) + \text{COMP}(\mathcal{G}) \tag{12}$$

where $f(\mathcal{D}_{train}, \mathcal{D}_g)$ is the divergence between the training data and generated data, COMP$(\mathcal{G})$ *is the parametric complexity in Eqn 11, and $\alpha > 0$ is a scalar that controls the relative importance between the divergence and the complexity.*

If $\mathcal{G}$ memorizes $\mathcal{D}_{train}$, then $f(\mathcal{D}_{train}, \mathcal{D}_g)$ goes to 0 but COMP$(\mathcal{G})$ blows up. If COMP$(\mathcal{G})$ goes to 0, then generated samples are concentrated in a small region, making $f(\mathcal{D}_{train}, \mathcal{D}_g)$ goes up (assuming that f is a divergence that is good at detecting the lack of diversity, e.g. our updated NND). Therefore, our metric cannot be fooled by training set memorization. In general, f_{gen} cannot be 0 because the two terms cannot be 0 at the same time. α can be removed if we are comparing models with the same divergence.

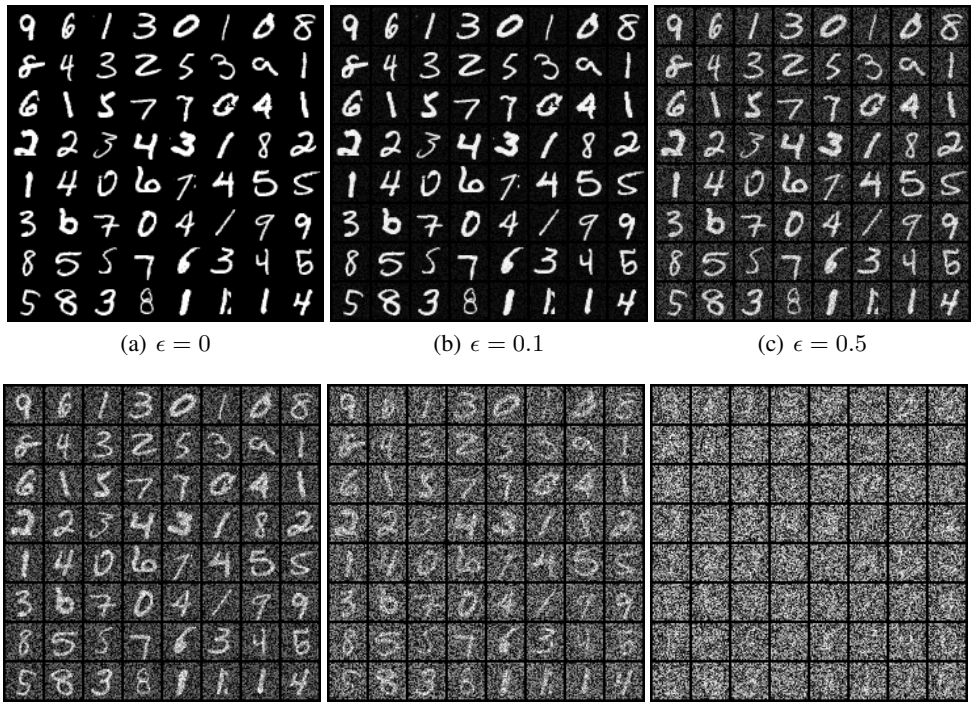
E.3 A constant speed regularizer for GANs

A DGLVM that produces constant speed paths in the space of generated data is desirable because 1) it is less likely to memorize the training set, 2) it does not make sudden jumps from one memorized datapoint to another, thus produces smoother interpolation. We propose the following constant speed regularizer for the generator $\mathcal{G}$ in GANs:

$$\mathcal{L}_{\mathcal{G}}^{\text{const}} = \mathcal{L}_{\mathcal{G}} + \lambda \mathbb{E}_{\boldsymbol{z}_i, \boldsymbol{z}_j \sim p_z} \left[\mathbb{E}_{t \in [0,1]} \left[\left(\left\| \frac{\partial \mathcal{G}(\boldsymbol{z}(t))}{\partial t} \right\| - \bar{s} \right)^2 \right] \right] \tag{13}$$

where $\mathcal{L}_{\mathcal{G}}$ is the original loss function of $\mathcal{G}$, $\bar{s}$ is the average speed on the path from $\mathcal{G}(\boldsymbol{z}_i)$ to $\mathcal{G}(\boldsymbol{z}_j)$, and the interpolation method is a constant speed interpolation method, i.e. $\left\| \frac{\partial \boldsymbol{z}}{\partial t} \right\| =$ const. The regularizer forces the variance of the speed to 0, making the speed constant. As noted by other authors (e.g. [16]), for natural image data like Imagenet, it might be better to apply the regularizer to the feature space. In the experiments below, we apply the regularizer directly to the data space because the dataset is simple.

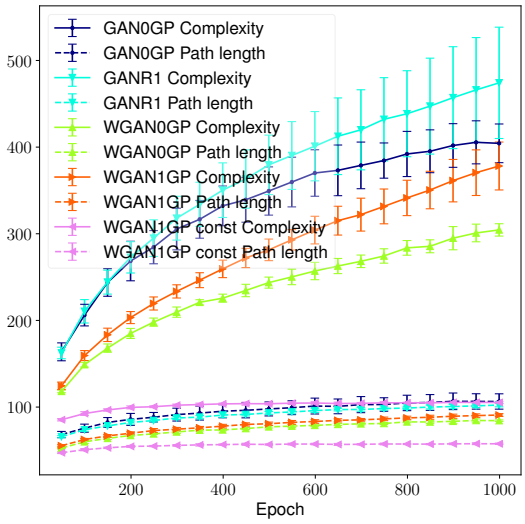
F Experiments



(a) $\epsilon = 0$ (b) $\epsilon = 0.1$ (c) $\epsilon = 0.5$

(d) $\epsilon = 1$ (e) $\epsilon = 2$ (f) $\epsilon = 5$

Figure 5: Noisy images generated by our procedure with $p_u = \mathcal{U}(-1, 1)$.



(a)

Figure 6: Experimental result for the computational complexity.