Toward a Generalization Metric for Deep Generative Latent Variable Models

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

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

Abstract

Deep Generative Latent Variable Models (DGLVMs) are generative models (GMs) that use deep neural networks to transform a latent distribution to another distribution that approximates a target distribution. The generalizability of a DGLVM depends on the generalizability of the generator network, which is very hard to measure correctly. Although a number of evaluation metrics for generative models have been proposed, they are not designed to measure generalization. In this paper, we develop a tool for comparing the ability to measure generalization of evaluation metrics. We introduce the concept of *breaking number* (BN) - the size of the smallest dataset that can fool an evaluation metric - and show that BNs correlate to the robustness of evaluation metrics. Inspired by the Minimum Description Length principle, we propose to measure the quality of DGLVMs in 2 parts: fitness and complexity. Experimental results show that our metric has better discriminative power and is more robust than existing metrics.

1 Introduction

Deep Generative Latent Variable Models (DGLVMs) such as GANs [9], VAEs [16], and RealNVP [8] use neural networks to transform a latent distribution p_z to a model distribution p_g that approximates a target distribution p_r. Although DGLVMs can produce high fidelity samples [6, 22], their ability to generalize outside of the training dataset is still an open question. A number of sample based evaluation metrics were proposed and compared against each other empirically (e.g. Inception Score (IS) [27], Frechet Inception Distance (FID) [13], k-means Precision-Recall (k-means PR) [26], and k-nearest neighbors Precision Recall (k-nn PR) [18]). However, it is still unclear that which one is better and whether a better score implies better generalization. Recently, [12] used the Neural Net Divergence (NND) [2] to estimate the generalizability of GANs. Experiments in the paper showed that if a GM only generates training datapoints, it cannot achieve a low NND on a test dataset. The authors concluded that in order to achieve a good NND on a test dataset, a GM needs to exhibit some level of generalization. However, as shown in Sec. 3.2, a generator with no real generalization capacity can achieve very low NND scores on both training and testing datasets. That raises the need for: 1) a method for comparing the capacity of evaluation metrics, and 2) a metric that can measure the generalizability of GMs.

In this paper, we develop a new tool for comparing the capacity of evaluation metrics in measuring generalization. We introduce the concept of *breaking number* (BN) of an evaluation metric - the size of the smallest dataset that achieves a better score than the training dataset. A metric with higher BN is more robust to training set memorization attack. We say that a metric can measure the generalizability of a GM if its BN is greater than the size of the training set. We construct the minimal datasets for the above metrics and find that their sizes are smaller than that of the training set. Thus, these metrics cannot be used to compare the generalizability of GMs (Sec. 3.2). We show that when a differentiable generator (e.g. a deep net) memorizes some training datapoints, Jacobian norm exploding occurs. To estimate the generalizability of a GM, we need to analyze the generated data and

Preprint. Under review.

the GM itself. Inspired by the Minimum Description Length principle [23], we evaluate DGLVMs using two factors: the divergence between generated data and training data, and the complexity of the model. We are interested in DGLVMs because they allow efficient generation of a path from one datapoint to another, which is then used to estimate the complexity of the model. We estimate the divergence using existing metrics and propose a new method for estimating the complexity of the DGLVM. A high quality, generalizable DGLVM should fit the training data well and have low complexity. Although our method can be applied to any DGLVMs, our experiments focus on GANs because they are one of the most common yet one of the least understood GMs.

Contributions

1. We introduce *breaking number* and compute BNs of common evaluation metrics (Sec. 3).
2. We propose a new method for estimating the complexity of DGLVMs and use it to compare the generalizability of DGLVMs (Sec. 4).
3. A new regularizer for improving the smoothness of the manifold generated by GANs.

2 Background and Related Work

2.1 Sample based evaluation metrics

The quality of a GM is defined by how close p_g is to p_r. The closeness is measured using a distance/divergence $d(p_r, p_g)$. The smaller $d(p_r, p_g)$ is, the closer p_g is to p_r. Because p_r (and usually p_g as well) is not available in closed form, the distance/divergence between the two distributions is estimated by the distance/divergence between the empirical versions of these distributions: $d(\hat{p}_r, \hat{p}_g)$ or $d(\mathcal{D}_r, \mathcal{D}_g)$ where $\hat{p}_r,\ \hat{p}_g$ are uniform distributions over the sets of real and generated samples $\mathcal{D}_r,\ \mathcal{D}_g$. From now, we write $d(\mathcal{D}_r, \mathcal{D}_g)$ because some metrics do not rely on distributions. Classic distances such as Wasserstein distance are intractable because they require an exponential number of samples to reliably estimate the distance. A number of evaluation metrics which use a polynomial number of samples have been proposed. We review here the most common ones.

Inception Score [27] is defined as

$$IS(\mathcal{D}_g) = \exp\big(\mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_g}[KL(p(y|\boldsymbol{x}) \,||\, p(y))]\big) \tag{1}$$

where $p(y|\boldsymbol{x})$ is the conditional distribution of the label given the input, $p(y) = \sum_{\boldsymbol{x}\in\mathcal{D}_g} p(y|\boldsymbol{x})p(\boldsymbol{x})$ is the marginal distribution of the labels of generated samples. $p(\boldsymbol{x}) = 1/|\mathcal{D}_g|$ and $p(y|\boldsymbol{x})$ is computed using a pretrained classifier, a common choice is the Inception model [28] trained on Imagenet [7]. Inception model can be replaced with other pretrained models to evaluate different datasets. IS involves only $\mathcal{D}_g$ because $\mathcal{D}_r$ is implicitly the training data (e.g. Imagenet). To fit the divergence minimization scheme, we define the 'divergence': $d_{IS} = k - IS$, where k is the number of classes in the training dataset. d_{IS} is minimized (IS is maximized) if the entropy H$(y|\boldsymbol{x})$ is 0, i.e. $\boldsymbol{x}$ perfectly resembles samples from Imagenet, and $p(y)$ is the uniform distribution over classes of Imagenet. A fundamental flaw of d_{IS}: a lower d_{IS} score implies that labels of generated samples are more uniformly distributed, not that $p(y)$ is closer to the true distribution over labels. (Un)Fortunately, the label distribution in Imagenet is uniform so d_{IS} can be used in practice.

Frechet Inception Distance [13] first approximates p_r and p_g using 2 Gaussians $\mathcal{N}(\boldsymbol{\mu}_r, \Sigma_r)$ and $\mathcal{N}(\boldsymbol{\mu}_g, \Sigma_g)$, then compute the Wasserstein-2 distance between them

$$d_{fid}(\mathcal{D}_r, \mathcal{D}_g) = \|\boldsymbol{\mu}_g - \boldsymbol{\mu}_r\|_2^2 + \text{Tr}(\Sigma_g + \Sigma_r - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{2}$$

where $\boldsymbol{\mu}_g, \Sigma_g$ and $\boldsymbol{\mu}_r, \Sigma_r$ are the mean, covariance matrix of $\mathcal{D}_g$ and $\mathcal{D}_r$, respectively. d_{fid} is minimized if $\boldsymbol{\mu}_g = \boldsymbol{\mu}_r$ and $\Sigma_g = \Sigma_r$. FID compares p_r and p_g using the first 2 moments so $d_{fid}(\mathcal{D}_r, \mathcal{D}_g) = 0$ does not guarantee that the two distributions are the same.

Neural Net Divergence (NND) [12, 2] is defined as

$$d_{nnd}(\mathcal{D}_r, \mathcal{D}_g) = \sup_{f_{\boldsymbol{\theta}}\in\mathcal{F}} (\mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_r}[f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_g}[f_{\boldsymbol{\theta}}(\boldsymbol{x})]) \tag{3}$$

where $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_x} \to \mathbb{R}$ is usually a *finite capacity* network with parameter $\boldsymbol{\theta}$. The supremum is found by training the network with variants of SGD. d_{nnd} is minimized if the two datasets are the same.

NND, FID, and IS use a scalar to measure both the quality and diversity of generated samples. The following metrics are designed to separate quality from diversity:

K-means Precision Recall [26] uses precision and recall to measure the quality and diversity of generated samples (Algo. 1). For $\beta > 1$ (the authors recommended $\beta = 8$), F_β and $F_{1/\beta}$ scores can be interpreted as the recall and precision of the model, respectively. k-means PR can distinguish models that generate high quality (high precision), low diversity (low recall) samples from models that generate low quality (low precision), high diversity (high recall) samples. We define the divergences: $d_{p-kmeans} = 1 - F_{1/\beta},\ d_{r-kmeans} = 1 - F_\beta$.

K-Nearest Neighbors Precision Recall [18] calculates precision and recall by comparing the 2 reconstructed manifolds (Algo. 2 and Fig. 1(a)-1(c)). The authors empirically showed that k-nn PR has better discriminative power than k-means PR on some datasets and models. However, as shown in Sec. 3.2, k-nn PR has a lower breaking number than k-means PR and thus is more susceptible to dataset memorization attack. We define the divergences: $d_{p-knn} = 1 - P,\ d_{r-knn} = 1 - R$.

2.2 Other related work

Arora et al. [3] used the birthday paradox and human evaluation to estimate the number of distinct samples that a GM can generate. The method cannot be automated and has high variance and bias because humans are involved in the loop. The authors of StyleGAN [14] 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. We show in Sec. 4 that the pairwise distance does not well reflect the smoothness of the manifold and it is minimized when total mode collapse [9] occurs. Therefore, pairwise distance is not a good metric for GANs and DGLVMs in general.

[4] showed that many distributions that are far from the target distribution p_r can achieve high IS scores but not higher than the training set. [12] empirically showed that a generator that memorizes a small subset of the training set can outperform a good GAN in terms of IS and FID scores. In Sec. 3.2, we construct the smallest dataset $\mathcal{D}^*$ that outperforms the training set and show the link between $\mathcal{D}^*$ and the robustness of evaluation metrics.

2.3 A GM with poor generalizability

Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ and $\boldsymbol{\epsilon} = [\epsilon_1, ..., \epsilon_n]^\top$,we define the following generative procedure:

1. Draw $\boldsymbol{x}_i$ from $\mathcal{D}$ according to a categorical distribution: $p(\boldsymbol{X} = \boldsymbol{x}_i|\boldsymbol{\mu}) = \mu_i,\ \sum_j \mu_j = 1$.
2. Draw a noise vector $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \epsilon_i I)$
3. Output the generated datapoint: $\boldsymbol{x} = \boldsymbol{x}_i + \boldsymbol{u}$

This generator, denoted as $\mathcal{G}_\mathcal{D}$, has no real generalization capacity. If $\boldsymbol{\epsilon} = \boldsymbol{0}$, it can only generate points from $\mathcal{D}$. If $\boldsymbol{\epsilon} > \boldsymbol{0}$, $\mathcal{G}_\mathcal{D}$ can be approximated using a finite capacity neural net. Using $\mathcal{G}_\mathcal{D}$, we can create datasets of arbitrary size while the support of $p_{\mathcal{G}_\mathcal{D}}$ is effectively $\mathcal{D}$. We will show that $\mathcal{G}_\mathcal{D}$ can fool all of the metrics in Sec. 2.1 even when $\mathcal{D}$ is a strict subset of the training dataset $\mathcal{D}_{train}$.

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 $F_\beta,\ F_{1/\beta}$

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 $P,\ R$

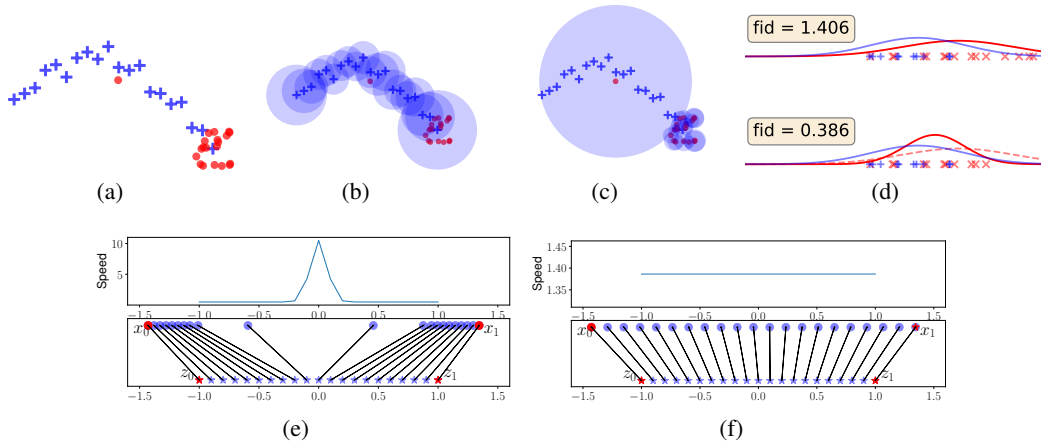

Figure 1: (a) real data (blue crosses) and fake data (red dots). (b), (c) real and fake manifolds approximated with 3-nearest neighbors. (d) top: original data and distributions, bottom: new red distribution with smaller support and lower FID. (e) a GM that memorizes the training set. (f) a GM that produces constant speed interpolation.

3 Breaking Number

3.1 Definition

If a GM memorizes the entire training set and output random samples from that set, the divergence between generated and training data goes to 0. To reduce the effect of training set memorization, the divergence should be computed between test data and generated data [12]. Inspired by *shattering number* [30], we introduce *breaking number*:

Definition 1. *Let $\mathcal{D}_{train} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m \sim p\}$, $\mathcal{D}_{test} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n \sim p\}$ be disjoint sets of i.i.d. samples from a distribution p with size m, n, respectively, and d be a divergence (evaluation metric). The Breaking Number of d (w.r.t. $\mathcal{D}_{train}$) is the size of the smallest set $\mathcal{D}^* = \{\boldsymbol{x}_1, ...\boldsymbol{x}_k \sim p \mid \boldsymbol{x}_i \notin \mathcal{D}_{test}\ \forall i = 1, ..., k\}$ s.t. $d(\mathcal{D}_{test}, \mathcal{D}^*) < d(\mathcal{D}_{test}, \mathcal{D}_{train})$*

$$BN(d) = |\mathcal{D}^*|.$$

$BN(d)$ depends on the specific $\mathcal{D}_{train}, \mathcal{D}_{test}$ used in computing the divergence. If m, n are large enough, we expect $BN(d)$ to stay the same for different realizations of $\mathcal{D}_{train}, \mathcal{D}_{test}$. If $BN(d) \leq |\mathcal{D}_{train}|$ then d cannot differentiate a model that actually generalizes from a model that simply memorizes training data. If $BN(d) > |\mathcal{D}_{train}|$ then to fool d, a GM must be able to generate more distinct samples than the training set. We say that the metric can be used to measure generalization. Some metrics like k-means PR implicitly require $|\mathcal{D}| = |\mathcal{D}_{test}|$ in order to compute $d(\mathcal{D}_{test}, \mathcal{D})$. A dataset of size $|\mathcal{D}_{test}|$ can be created from $\mathcal{D}^*$ by duplicating some of its elements or using the generator $\mathcal{G}_{\mathcal{D}^*}$ to generate noisy versions of some datapoints. In such cases, the BN is still $|\mathcal{D}^*|$, not $|\mathcal{D}_{test}|$. We note that [12] proposed a definition of a model outperforming training set memorization. However, the definition contains several weaknesses which we discuss in Appendix A.

3.2 Breaking sample based evaluation metrics

***Inception Score**:* Because IS always compares the generated data against $\mathcal{D}_{train}$ - the dataset on which the classifier was trained - we cannot apply the above definition of BN to IS. However, we can still construct the smallest set that maximizes IS. Assume that the classifier is perfect, then $p(y|\boldsymbol{x}_i) = \mathbf{1}_{y=y_i},\ \forall \boldsymbol{x}_i \in \mathcal{D}_{train}$. $\mathcal{D}^*_{IS}$ is constructed by picking 1 sample from each of the k classes of $\mathcal{D}_r$: $\mathcal{D}^*_{IS} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_k | \boldsymbol{x}_i \in \mathcal{D}_{train},\ y_i = i, \forall i = 1, ..., k\}$. The marginal distribution is $p(y = i) = 1/k,\ \forall i$. Plug this in Eqn. 1, we have $IS(\mathcal{D}^*_{IS}) = k$ and $d_{IS}(\mathcal{D}^*_{IS}) = 0$. Thus, $BN(d_{IS}) = k$. In the following, we assume that $d(\mathcal{D}_{test}, \mathcal{D}_{train}) > 0$.

***Frechet Inception Distance**:*

Proposition 1. *For large enough $\mathcal{D}_{train},\ \mathcal{D}_{test}$, there exist a strict subset $\mathcal{D}' \subset \mathcal{D}_{train}$ s.t. $d_{fid}(\mathcal{D}_{test}, \mathcal{D}') < d_{fid}(\mathcal{D}_{test}, \mathcal{D}_{train})$.*

Proof. See Appendix B. Fig. 1(d) shows an example on 1-dimensional datasets. □

Because $BN(d_{fid}) \leq |\mathcal{D}'| < |\mathcal{D}_{train}|$, FID cannot measure generalization. To estimate $BN(d_{fid})$, we can iterate through the powerset of $\mathcal{D}_{train}$ to find smallest subset of $\mathcal{D}_{train}$ satisfying Def. 1.

Neural Net Divergence's breaking number depends on the network and the training algorithm so it cannot be computed theoretically. We design the following experiment to estimate $BN(d_{nnd})$: Let $\mathcal{D}_{train},\ \mathcal{D}_{test}$ be 2 disjoint subsets of the MNIST dataset [19] with $|\mathcal{D}_{train}| = |\mathcal{D}_{test}| = 10000$, and $\mathcal{D}'_{train} \subset \mathcal{D}_{train}$ with $|\mathcal{D}'_{train}| = |\mathcal{D}_{train}|/2$. Fake datapoints are generated continuously through out the training process using a generator $\mathcal{G}_{\mathcal{D}'_{train}}$ with $\boldsymbol{\epsilon} = 0.1$. Let $\mathcal{D}_g$ be the set of generated datapoints. The network is a MLP and is trained for 100 epochs. Implementation details are in Appendix C. The averaged result after 5 runs is: $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_{train}) = 0.52 \pm 0.01$ and $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g) = 0.312 \pm 0.013$. $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g)$ decreases as $\boldsymbol{\epsilon}$ and/or $|\mathcal{D}_{train}|$ increase, and reaches 0.049 ± 0.002 when $\mathcal{D}'_{train} = \mathcal{D}_{train}$ and $\boldsymbol{\epsilon} = 1$. $\mathcal{D}_g$ outperforms $\mathcal{D}_{train}$ while having an effective support of size $|\mathcal{D}_{train}|/2$. However, if $|\mathcal{D}'_{train}| = |\mathcal{D}_{train}|/4$ then $\mathcal{D}_g$ cannot outperform $\mathcal{D}_{train}$. Thus, for this setting, $2500 < BN(d_{nnd}) < 5000$. The experiment shows that $BN(d_{nnd}) < |\mathcal{D}_{train}|$. It also suggests that the GANs in [12] might not really generalizes even though they outperformed training set memorization.

The above experiment exploits a flaw in the definition of NND in [12]: there are no restrictions on the sizes of datasets. Because fake data is generated continuously during training, $|\mathcal{D}_g|$ is much larger than $|\mathcal{D}_{train}|$ and $\mathcal{D}_g$ covers a larger region of the data space. Therefore, maximizing $(\mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_{test}}[f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_g}[f_{\boldsymbol{\theta}}(\boldsymbol{x})])$ is harder than maximizing $(\mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_{test}}[f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_{train}}[f_{\boldsymbol{\theta}}(\boldsymbol{x})])$. We propose to fix NND by requiring $|\mathcal{D}_g| = |\mathcal{D}_{test}|$. With this requirement, to compute NND of a GM, one need to sample a dataset $\mathcal{D}_g$ of size $|\mathcal{D}_{test}|$, then compute $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g)$ using Eqn. 3.

To fool Precision-Recall metrics, a dataset must simultaneously maximize precision and recall (minimize the corresponding divergences).

K-means Precision-Recall: Let $\mathcal{C} \leftarrow kmeans(\mathcal{D}_{train} \cup \mathcal{D}_{test})$ be the resulting clusters of k-means algorithm. If $\mathcal{D}_{train},\ \mathcal{D}_{test}$ are large enough, then in every cluster $\mathcal{C}_i$, there must be some training and some testing datapoints. Intuitively, $\mathcal{D}^*_{kmeans}$ can be constructed by picking 1 training datapoint from each cluster and duplicating it $|\mathcal{C}_{i,test}|$ times, where $\mathcal{C}_{i,test}$ is the set of testing datapoints in cluster $\mathcal{C}_i$. See Appendix D for details of the construction algorithm. Plugging $\mathcal{D}_{test},\ \mathcal{D}^*_{kmeans}$ to Algo. 1, we see that the 2 histograms are identical so $F_\beta = F_{1/\beta} = 1$ for every β. The breaking number of k-means PR is k - the number of clusters - which is smaller than $|\mathcal{D}_{train}|$.

K-Nearest Neighbors Precision-Recall has a constant breaking number of 2. If $\mathcal{D}_{train}, \mathcal{D}_{test}$ are large enough, there exist 2 points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{M}_{train} \cap \mathcal{M}_{test}$ s.t. $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \|\boldsymbol{x}_i - \boldsymbol{x}_k\|,\ \forall \boldsymbol{x}_k \in \mathcal{D}_{train} \cup \mathcal{D}_{test}$. The fake data set is created by $\boldsymbol{x}_i$ and $|\mathcal{D}_{test}| - 1$ duplicates/noisy versions of $\boldsymbol{x}_j$. Because $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{M}_{test}$, the precision is 1. Because the hypersphere with center $\boldsymbol{x}_i$ and radius $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ contains all points in $\mathcal{D}_{train} \cup \mathcal{D}_{test}$, the recall is 1. An example on a 2-dimensional dataset is shown in Fig. 1(c): although there are only 2 significantly different datapoints (clusters), the fake manifold still covers all real datapoints and the real manifold also covers all of the fake datapoints. More experiments and discussion are located in Appendix E.

Discussion: To achieve good k-means PR, IS, FID, and NND scores, the fake dataset must match some statistics of the real dataset. That requires the GM to generate samples belong to different classes/clusters. A GM with mode collapse can hardly achieve good scores in these metrics. To break k-nn PR, only 2 samples are needed and their labels are not important. A GM with mode collapse can still achieve high k-nn PR. For k-nn PR to work well, the generated data should not contain outliers like the one in Fig. 1(a). In Sec. 5, we experimentally verify that statement on real world datasets.

4 An MDL inspired Generalization Metric

4.1 Motivation

4.1.1 Preliminaries

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 $\boldsymbol{x}_1 = \mathcal{G}(\boldsymbol{z}_1)$: $\boldsymbol{x} : [0,1] \to \mathbb{R}^{d_x},\ \boldsymbol{x}(t) = \mathcal{G}(\boldsymbol{z}(t))$. The length of the path from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$ is

$$\ell(\boldsymbol{x}_0, \boldsymbol{x}_1) = \int_0^1 \left\| \frac{\partial \boldsymbol{x}}{\partial t} \right\|_F dt = \int_0^1 \left\| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial t} \right\|_F dt \tag{4}$$

Minimum Description Length principle [17, 23, 24, 25, 5] (see [10] 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

$$\text{SCOMP}(\mathcal{D}|\mathcal{H}) = \text{LEN}(\mathcal{D}|\hat{H}) + \text{COMP}(\mathcal{H}) \tag{5}$$

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. 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 [10]). L and $\text{COMP}(\mathcal{H}_L)$ are positively correlated. In the following, we use L in place of the parametric complexity of $\mathcal{H}_L$.

4.1.2 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. 1(e) 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. 1(e)). Fig. 1(f) 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. 1(e) is much higher than that in Fig. 1(f), the paths in the two figures have the same length. The maximum speed is a better feature for detecting memorization than the path length (Sec. 5).

From Eqn. 4, 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

$$\text{COMP}(\mathcal{G}) = \mathbb{E}_{\boldsymbol{z}_i, \boldsymbol{z}_j \sim p_z}[s_{max}(\mathcal{G}(\boldsymbol{z}_i), \mathcal{G}(\boldsymbol{z}_j))] \tag{6}$$

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 (Appendix F).

4.2 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 2. *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:*

$$d_{gen}(\mathcal{G}) = \alpha d(\mathcal{D}_{train}, \mathcal{D}_g) + \textit{COMP}(\mathcal{G}) \tag{7}$$

where $d(\mathcal{D}_{train}, \mathcal{D}_g)$ is the divergence between the training data and generated data, COMP$(\mathcal{G})$ is the parametric complexity in Eqn 6, 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 $d(\mathcal{D}_{train}, \mathcal{D}_g)$ goes to 0 but $\text{COMP}(\mathcal{G})$ blows up. If $\text{COMP}(\mathcal{G})$ goes to 0, then generated samples are concentrated in a small region, making $d(\mathcal{D}_{train}, \mathcal{D}_g)$ goes up (assuming that d is a divergence that is good at detecting the lack of diversity, e.g. our modified d_{nnd}). Therefore, our metric cannot be fooled by training set memorization. In general, d_{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.

4.3 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:

$$\mathcal{L}_{\mathcal{G}}^{\text{const}} = \mathcal{L}_{\mathcal{G}} + \lambda \mathbb{E}_{\boldsymbol{z}_i, \boldsymbol{z}_j \sim p_z} \left[\mathbb{E}_{t \in [0,1]} \left[\left(\left\| \frac{\partial \mathcal{G}(\boldsymbol{z}(t))}{\partial t} \right\| - \bar{s} \right)^2 \right] \right] \tag{8}$$

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. $\left\| \frac{\partial \boldsymbol{z}}{\partial t} \right\| = \text{const}$. The regularizer forces the variance of the speed to 0, making the speed constant. As noted by other authors (e.g. [14]), 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.

5 Experiments

5.1 Evaluating evaluation metrics

We evaluate k-means PR, k-nn PR, our modified NND and our generalization metric on MNIST dataset.[1] We set $|\mathcal{D}_{train}| = |\mathcal{D}_{test}| = |\mathcal{D}_g| = 10000$. Implementation deails and results on CelebA [20] are located in Appendix G. We trained Wasserstein GAN with 1 centered Gradient Penalty (WGAN1GP) [1, 11], WGAN with 0 centered GP (WGAN0GP) [29, 32], GAN with 0 centered GP (GAN0GP) [29], GAN with R1 GP (GANR1) [21], and WGAN1GP with constant speed regularizer with $\lambda = 0.01$ (WGAN1GP-const), for 1000 epochs and averaged the results of 5 runs. We trained 2 groups of models: group 1 was trained with learning rate $lr_1 = 2 \times 10^{-4}$, group 2 was trained with $lr_2 = 10^{-3}$. Group 1 produced diverse, high quality samples. The GAN0GP, GANR1 instances in the group 2 produced bad samples (Fig. 2(e)). We use spherical interpolation [31] (a constant speed interpolation method if $\|\boldsymbol{z}_i\| = \|\boldsymbol{z}_j\|$) to generate the interpolated samples in Fig. 2(f), 2(g).

***k*-means PR** (Fig. 2(a)) has low discriminative power as all models, except WGAN1GP-const, achieved almost perfect F_β and $F_{1/\beta}$ scores. This agrees with the result reported in [18].

***k*-nn PR**: For good GANs, k-nn PR has better discriminative power than k-means. This is shown by the clearer difference between models in Fig. 2(b). However, k-nn PR fails to recognize that the model in Fig. 2(e) is a bad model. The model that generated the samples in Fig. 2(e) has $P = 0.64, R = 1.00$. The recall also have a very high variance as shown in the top pane in Fig. 2(e). In contrast, other metrics can recognize that the model is bad. The model's scores in other metrics are much worse than that of models in the first group: $F_\beta = 0.56$, $F_{1/\beta} = 0.63$, $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g) = 15.46$ (full result in Appendix G). k-nn PR can evaluate reasonably good GMs but it cannot evaluate bad ones.

Modified NND: Fig. 2(c) shows that none of the model can outperform training set memorization.[2] The high NND may due to the low quality of generated samples, it does not necessary mean that the models did not generalize. The discriminative power of NND is low as most models have roughly the same NND. The exception is WGAN0GP which has the lowest NND. Fig. 2(c) shows that models trained with 0GP have lower $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g)$ than models trained with 1GP or R1. That confirms the claim in [29] that 0GP improves the generalizability of GANs.

Parametric Complexity (PC): Fig. 2(d) shows that PC can clearly differentiate different models. In contrast, the average path length cannot distinguish WGAN1GP from WGAN0GP or GAN0GP from GANR1. Similar to NND, PC also shows that 0GP significantly reduces the complexity of the generator (although it is applied to the discriminator). WGAN1GP-const has the lowest PC and average path length and as shown in the next subsection, it has the smoothest interpolation path. The result shows that PC is predictive of memorization behaviors.

[1] IS and FID are not included because they are mostly used for Imagenet like datasets.

[2] If $|\mathcal{D}_{test}| > |\mathcal{D}_{train}|$ and the network used for computing NND has enough capacity, our modified NND might have a breaking number greater than $|\mathcal{D}_{train}|$. We leave the answer for that for future work.

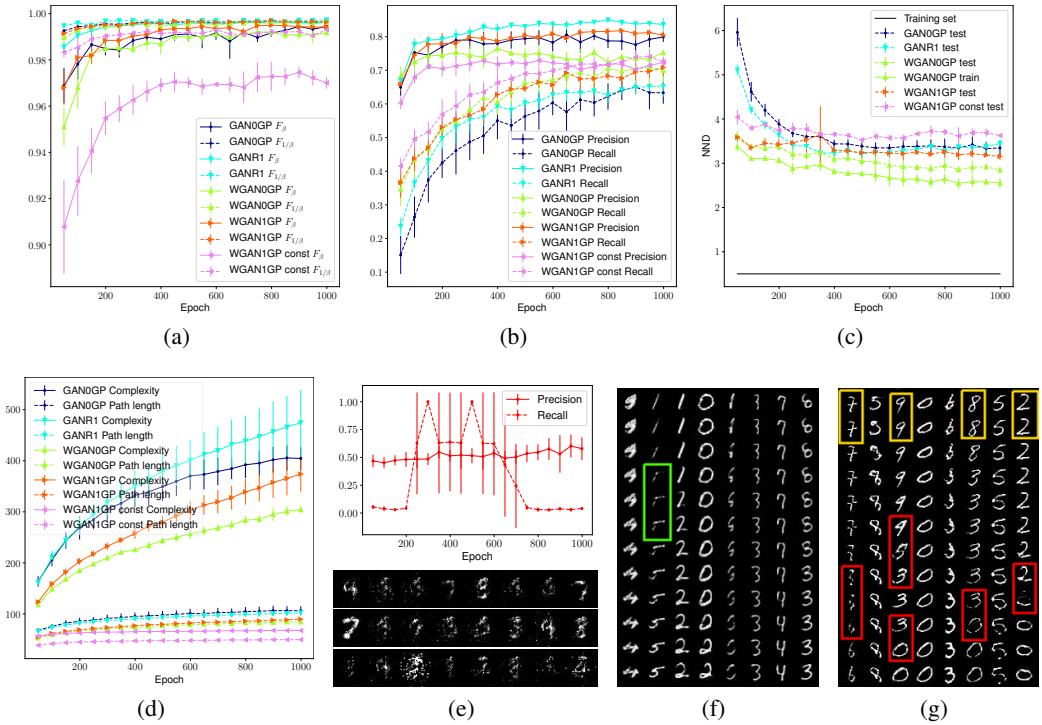

Figure 2: (a) - (d) results of k-means PR, k-nn PR, NND, and COMP($\mathcal{G}$) and average path length of models trained with $lr_1 = 2 \times 10^{-4}$. (c) Because $d_{nnd}(\mathcal{D}_{train}, \mathcal{D}_g)$ and $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g)$ always show the same trend, we show only $d_{nnd}(\mathcal{D}_{test}, \mathcal{D}_g)$. We show $d_{nnd}(\mathcal{D}_{train}, \mathcal{D}_g)$ for WGAN0GP as a representative. (e) top: k-nn PR of GAN0GP trained with $lr_2 = 10^{-3}$, bottom: generated samples. (f), (g) interpolation result of WGAN1GP-const and WGAN1GP at epoch 1000.

5.2 Constant speed regularizer

Fig. 2(g) shows that WGAN1GP usually makes little changes in the first interpolation steps (yellow boxes) and large jumps in the middle of the path (red boxes). The phenomenon suggests that WGAN1GP is memorizing the training set (similar to Fig. 1(e)). The behavior agrees with the large parametric complexity of WGAN1GP in Fig. 2(d). Other models with large parametric complexity also exhibit the same behaviors (Appendix G).

WGAN1GP-const produces smooth interpolation and consecutive steps show constant changes (Fig. 2(f)). In Fig. 2(d), the average max speed (complexity) curve is very close to the average path length curve. It implies that for most paths, the max speed is close to the average speed, i.e. the paths have near constant speeds. One exception is the digits in the green box. We believe that this is due to the the large difference between the digits at the two end points. Constant speed interpolation comes at a cost of worse scores in other metrics (Fig. 2(a), 2(b), 2(c)). Better balance between smoothness and fitness is achieved with smaller λ (Appendix G).

6 Conclusion

In this paper, we study the ability to measure generalization of different evaluation metrics for generative models. We introduce *breaking number* - a theoretical tool for comparing the capacity of evaluation metrics. Using *breaking number*, we prove that all of the previously invented evaluation metrics cannot measure generalization. We propose a generalization metric for DGLVMs which is robust against training set memorization attack. Our metric shows better discriminative power than previous metrics and can effectively detect memorization behavior in DGLVMs. Our constant speed regularizer effectively helps GANs to improve the smoothness of the generated manifold and avoid training set memorization.

Statement of Broader Impact

Deep generative models like GANs can simulate realistic photos and other modalities. This capability has enabled novel applications in various areas such as fashion, arts, cartoons, drug discovery, physical simulation and privacy-preserving data sharing. However, much less is known about how truly realistic and representative the generated data is. Research in the past years has introduced multiple metrics to measure the quality of generative methods, but none has proved to be truly effective. This may lead to overconfidence in running unreliable data simulation that does not reflect the true nature of the real data, possibly causing wrong conclusions if the simulated data is used for decision making.

We address this problem by developing new mathematical tools for comparing the capacity of evaluation metrics, and for measuring generalization. First, we introduce a new concept of *breaking number* (BN) of an evaluation metric - the size of the smallest dataset that achieves a better score than the training dataset. A metric with higher BN is more robust to training dataset memorization attack. We construct the minimal datasets for the common evaluation metrics and find that all of them have low BNs. Second, inspired by the celebrated Minimum Description Length principle, we propose a metric for generalization which consists of two sub-metrics: the *divergence* between the generated data and the training data, and the *complexity* of the model. Third, we propose a new method for estimating the complexity of deep generative latent variable models.

Our results contribute to the advance of AI and the society at large:

- **Potential benefits to the society**: The wide spread use of deep generative models in fashion, arts, drug discovery and data simulation opens up new ways for creativity and scientific discovery. In areas where simulated data need to be truly reflective of the real data, we need good generalization metrics to quantify the fidelity and reliability of the data. Our solution offers one step toward trustable AI, which will boost the wider adoption with less negative consequences.
- **Advancing AI**: Our solutions push the frontiers of generative AI, an emerging sub-field that is open for deep theoretical advancements. We call for rethinking of evaluation metrics. In particular, we shown that existing generalization metrics do not live up to their expectation. This is achieved through a novel concept of *Breaking Number*. We then proposed a new metric that exhibits a much better capacity to measure generalization. Finally, we contributed a new regularizer for improving the smoothness of the manifold generated by generative models.

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] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[7] 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.

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

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

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

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

[18] 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.

[19] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[20] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset.

[21] 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.

[22] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *CoRR*, abs/1906.00446, 2019.

[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.

[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] 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`.

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

[30] V Vapnik and Y Alexey. Chervonenkis (1964).“on the uniform convergence of relative frequencies of events to their probabilities,”. *Theory of Probability and its Applications, v16 (2)*, pages 264–280.

[31] Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

[32] Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. Wasserstein divergence for gans. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 653–668, 2018.