A Non-Parametric Test to Detect Data-Copying in Generative Models

Casey Meehan Kamalika Chaudhuri Sanjoy Dasgupta
University of California, San Diego

Abstract

Detecting overfitting in generative models is an important challenge in machine learning. In this work, we formalize a form of overfitting that we call *data-copying* – where the generative model memorizes and outputs training samples or small variations thereof. We provide a three sample test for detecting data-copying that uses the training set, a separate sample from the target distribution, and a generated sample from the model, and study the performance of our test on several canonical models and datasets.

1 Introduction

Overfitting is a basic stumbling block of any learning process. While it has been studied in great detail in the context of supervised learning, it has received much less attention in the unsupervised setting, despite being just as much of a problem.

To start with a simple example, consider a classical kernel density estimator (KDE), which given data $x_1, \ldots, x_n \in \mathbb{R}^d$, constructs a distribution over $\mathbb{R}^d$ by placing a Gaussian of width $\sigma > 0$ at each of these points, yielding the density

$$q(x) = \frac{1}{(2\pi)^{d/2}\sigma^d n} \sum_{i=1}^{n} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right). \tag{1}$$

The only parameter is the scalar σ. Setting it too small makes $q(x)$ too concentrated around the given points: a clear case of overfitting (see Appendix **Figure 6**). This cannot be avoided by choosing the σ that maximizes the log likelihood on the training data, since in the limit $\sigma \to 0$, this likelihood goes to ∞. One solution here is to instead maximize log-likelihood on a held-out validation set.

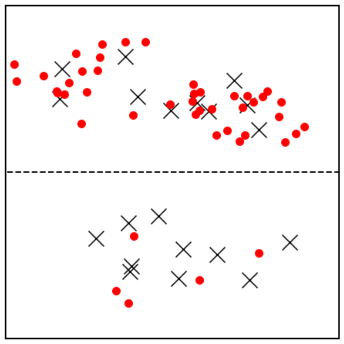
Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

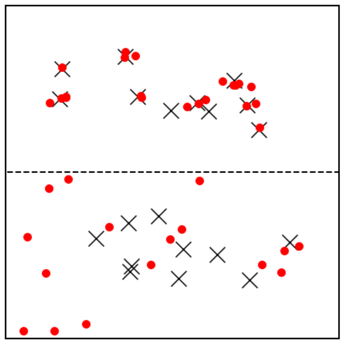
If even the one-parameter kernel density estimator is capable of overfitting, the situation is truly treacherous for the enormously complex generative models that have emerged over the past decade or so. Variational Auto Encoders (VAEs) (Kingma and Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) for example can easily involve millions of parameters. A major challenge in evaluating overfitting in these models is that most do not offer exact, tractable likelihoods. VAEs can only tractably provide a likelihood lower bound, while GANs have no accompanying density estimate at all. Thus any method that can assess these generative models must be based only on samples produced by them.

A body of prior work has provided tests for evaluating generative models based on samples drawn from them (Salimans et al., 2016; Sajjadi et al., 2018; Wu et al., 2017; Heusel et al., 2017); however, the vast majority of these tests focus on 'mode dropping' and 'mode collapse': the tendency for a generative model to either merge or delete high-density modes of the true distribution. A generative model that simply reproduces the training set or minor variations thereof will pass most of these tests.

In contrast, in this work we formalize and investigate a particular type of overfitting that we call 'data-copying': the propensity of a generative model to recreate minute variations of a subset of training examples it has seen, rather than represent the true diversity of the data distribution. An example is shown in **Figure 1b**; in the top region of the instance space, the generative model data-copies, or creates samples that are very close to the training samples; meanwhile, in the bottom region, it underfits. To detect this, we introduce a data-copying hypothesis test that relies on three independent samples: the original training sample used to produce the generative model; a separate (held-out) test sample from the underlying distribution; and a synthetic sample drawn from the generator.

Our key insight is that an overfit generative model would produce samples that are too close to the training samples – closer on average than an independently drawn test sample from the same distribution. Thus,


(a) Illustration of over-/under-representation
Training sample: $\times$, Generated sample: ●


(b) Illustration of data-copying/underfitting
Training sample: $\times$, Generated sample: ●

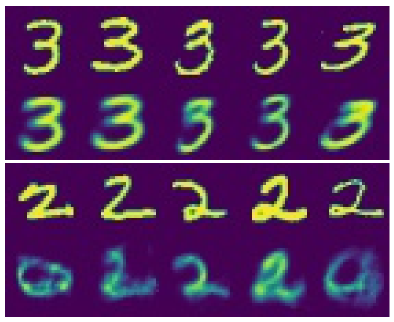

(c) VAE copying/underfitting on MNIST
top: $Z_U = -8.54$, bottom: $Z_U = +3.30$

Figure 1: Comparison of data-copying with over/under representation. Each image depicts a single instance space partitioned into two regions. Illustration **(a)** depicts an over-represented region (top) and under-represented region (bottom). This is the kind of overfitting evaluated by methods like FID score and Precision and Recall. Illustration **(b)** depicts a data-copied region (top) and underfit region (bottom). This is the type of overfitting focused on in this work. Figure **(c)** shows VAE-generated and training samples from a data-copied (top) and underfit (bottom) region of the MNIST instance space. In each 10-image strip, the bottom row provides random generated samples from the region and the top row shows their training nearest neighbors. Samples in the bottom region are on average further to their training nearest neighbor than held-out test samples in the region, and samples in the top region are closer, and thus 'copying' (computed in embedded space, see Experiments section).

if a suitable distance function is available, then we can test for data-copying by testing whether the distances to the closest point in the training sample are on average smaller for the generated sample than for the test sample. A further complication is that modern generative models tend to behave differently in different regions of space; a configuration as in **Figure 1b** for example could cause a global test to fail. To address this, we use ideas from the design of non-parametric methods by dividing the instance space into cells, conducting our test separately in each cell, and then combining the results to get a sense of the average degree of data-copying.

1.1 Related work

There has been a large body of prior work on the evaluation of generative models (Salimans et al., 2016; Lopez-Paz and Oquab, 2016; Richardson and Weiss, 2018; Sajjadi et al., 2018; Xu et al., 2018; Wu et al., 2017) . However, most are geared to detect some form of mode-collapse or mode-dropping: the tendency to either merge or delete high-density regions of the training data. Consequently, they fail to detect even the simplest case of extreme data-copying – where a generative model memorizes and exactly reproduces a bootstrap sample from the training set. We discuss below a few of these tests that use ideas similar to ours.

To-date there is a wealth of techniques for evaluating whether a model mode-drops or -collapses. Tests like the popular Inception Score (IS), Frechét Inception Distance (FID) (Heusel et al., 2017), Precision and Recall test (Sajjadi et al., 2018), and extensions thereof (Kynkäänniemi et al., 2019; Che et al., 2016) all work by embedding samples using the features of a discriminative network like 'InceptionV3' and somehow checking whether the training and generated distributions are similar *in aggregate*. The hypothesis-testing binning method proposed by Richardson and Weiss (2018) also compares aggregate population data, without embedding samples. The parametric Kernel MMD method proposed by Sutherland et al. (2016) uses a carefully selected kernel to estimate the distribution of both the generated and training samples and reports the maximum mean discrepancy between the two. Such tests will reward a generative model that only produces slight variations of the training set, since a 'good' aggregate distribution is defined by the training sample.

Lopez-Paz and Oquab (2016) propose the *Two-Sample Nearest Neighbor*, a two-sample non-parametric test. Their method groups a training and generated sample of equal cardinality together, with training points labeled '1' and generated points labeled '0', and then reports the Leave-One-Out (LOO) Nearest-Neighbor (NN) accuracy of predicting '1's and '0's. We report two values as discussed by Xu et al. (2018): the LOO accuracy of the training points, and the LOO accuracy of the generated points. An ideal generative model should produce an accuracy of 0.5 for each sample. Unlike this method, our test not only detects exact data-copying, which is unlikely, but estimates whether a given model generates samples closer to the training set than it should, as determined by a held-out test set.

The concept of data-copying has been explored by Xu et al. (2018) (where it is called 'memorization') for a variety of generative models and several of the above two-sample evaluation tests. Their results indicate that only the two-sample NN test is able to capture instances of extreme data-copying. Similarly, Bounliphone et al. (2016) explores three-sample testing, but for comparing the performance of different models, not for detecting overfitting. Other works contemporary with this take parametric approaches to finding data copying via neural network divergences (Gulrajani et al., 2020) and via learning reverse mappings for latent code models (Webster et al., 2019). The present work departs from those by offering a probabilistically motivated non-parametric test that is entirely model agnostic.

2 A Hypothesis Test for Data Copying

Let $\mathcal{X}$ denote an instance space in which data points lie, and P an unknown underlying distribution on this space. A training set T is drawn from P and is used to select a generative model Q. We then wish to assess whether Q is the result of overfitting: that is, whether Q is too close to the training data. To help ascertain this, we are able to draw two additional samples: a fresh sample of n points from P called P_n; a sample of m points from Q; called this Q_m.

As illustrated in **Figures 1a, 1b**, generative model overfitting can occur locally in some subspace $\mathcal{C} \subseteq \mathcal{X}$. To characterize this for any distribution D on $\mathcal{X}$, let $D|_\mathcal{C}$ denote its restriction to the region $\mathcal{C}$, that is,

$$D|_\mathcal{C}(\mathcal{A}) = \frac{D(\mathcal{A} \cap \mathcal{C})}{D(\mathcal{C})} \quad \text{for any } \mathcal{A} \subseteq \mathcal{X}.$$

2.1 Types of Overfitting

We now formalize the distinction between data-copying and over-representation, starting with a probabilistic interpretation of data-copying.

Intuitively, overfitting refers to situations where Q is "too close" to the training set T; that is, closer to T than the target distribution P happens to be. To make this quantitative, we begin by choosing a distance function $d : \mathcal{X} \to \mathbb{R}$ from points in $\mathcal{X}$ to the training set, for instance, $d(x) = \min_{t \in T} \|x - t\|^2$, if $\mathcal{X}$ is a subset of Euclidean space.

Naturally, we desire that Q's expected distance to the training set is the same as that of P's, namely $\mathbb{E}_{X \sim P}[d(X)] = \mathbb{E}_{Y \sim Q}[d(Y)]$. We may rewrite this as follows: given any distribution D over $\mathcal{X}$, define $L(D)$ to be the one-dimensional distribution of $d(X)$ for $X \sim D$. We consider overfitting to have occurred if random draws from $L(P)$ are systematically larger than from $L(Q)$. The above equalized expected distance condition can be redrafted as

$$\mathbb{E}_{Y \sim Q}[d(Y)] - \mathbb{E}_{X \sim P}[d(X)] = \mathbb{E}_{\substack{A \sim L(P) \\ B \sim L(Q)}}[B - A] = 0 \tag{2}$$

However, we are less interested in how *large* the difference is, and more so how *often* B is larger than A. Let

$$\Delta_T(P, Q) = \Pr\left(B > A \mid B \sim L(Q), A \sim L(P)\right)$$

where $0 \leq \Delta_T(P, Q) \leq 1$ represents how 'far' Q is from training sample T as compared to true distribution P. A more interpretable yet equally meaningful condition is

$$\Delta_T(P, Q) = \mathbb{E}_{\substack{A \sim L(P) \\ B \sim L(Q)}}[\mathbb{1}_{B > A}] \approx \frac{1}{2}$$

which guarantees Equation 2 if densities $L(P)$ and $L(Q)$ have the same shape, but are possibly mean-shifted.

If $\Delta_T(P, Q) << \frac{1}{2}$, Q is data-copying training set T, since samples from Q are systematically closer to T than are samples from P. However, even if $\Delta_T(P, Q) \geq \frac{1}{2}$, Q may still be data-copying T. As exhibited in **Figures 1b** and **1c**, a model Q may data-copy in one region and underfit in others. In this case, Q may be further from T than is P *globally*, but much closer to T *locally*. As such, we consider Q to be data-copying if it is overfit in any subspace $\mathcal{C} \subseteq \mathcal{X}$:

Definition 2.1. A generative model Q is *data-copying* training set T if, in some region $\mathcal{C} \subseteq \mathcal{X}$, it is systematically closer to T by distance metric $d : \mathcal{X} \to \mathbb{R}$ than are samples from P . Specifically, if

$$\Delta_T(P|_\mathcal{C}, Q|_\mathcal{C}) < \frac{1}{2}$$

This type of overfitting is orthogonal to the type of overfitting addressed by many previous works (Heusel et al., 2017; Sajjadi et al., 2018), which we call 'over-representation'. Here, Q overemphasizes some region of the instance space $\mathcal{C} \subseteq \mathcal{X}$, often a region of high density in the training set T.

Definition 2.2. A generative model Q is *over-representing* P in some region $\mathcal{C} \subseteq \mathcal{X}$, if the probability of drawing $Y \sim Q$ is much greater than it is of drawing $X \sim P$. Specifically, if

$$Q(\mathcal{C}) - P(\mathcal{C}) \gg 0$$

In this work we focus on data-copying, and provide a novel test to detect it.

2.2 A Global Data-Copying Test

We introduce our data-copying test in the global setting, when $\mathcal{C} = \mathcal{X}$. Here, we have a null hypothesis H_0 suggesting that Q may equal P:

$$H_0 : \Delta_T(P, Q) \ = \ \frac{1}{2} \tag{3}$$

There are well-established non-parametric tests for this hypothesis, such as the Mann-Whitney U test (Mann and Whitney, 1947). Let $A_i \sim L(P_n), B_j \sim L(Q_m)$ be samples of $L(P), L(Q)$ given by P_n, Q_m and their distances $d(X)$ to training set T. The U statistic estimates the probability in Equation 3 by measuring the number of all mn pairwise comparisons in which $B_j > A_i$. An efficient and simple method to gather and interpret this test is as follows:

1. Sort the $n+m$ values $L(P_n) \cup L(Q_m)$ such that each instance $l_i \in L(P_n), l_j \in L(Q_m)$ has rank $R(l_i), R(l_j)$, starting from rank 1, and ending with rank $n+m$. $L(P_n), L(Q_m)$ have no tied ranks with probability 1 assuming their distributions are continuous.

2. Calculate the rank-sum for $L(Q_m)$ denoted R_{Q_m}, and its U score denoted U_{Q_m}:

$$R_{Q_m} = \sum_{l_j \in L(Q_m)} R(l_j), \quad U_{Q_m} = R_{Q_m} - \frac{m(m+1)}{2}$$

   Consequently, $U_{Q_m} = \sum_{ij} \mathbb{1}_{B_j > A_i}$.

3. Under H_0, U_{Q_m} is approximately normally distributed with > 20 samples in both $L(Q_m)$ and $L(P_n)$, allowing for the following z-scored statistic:

$$Z_U\big(L(P_n), L(Q_m); T\big) = \frac{U_{Q_m} - \mu_U}{\sigma_U},$$
$$\mu_U = \frac{mn}{2}, \quad \sigma_U = \sqrt{\frac{mn(m+n+1)}{12}}$$

Z_U provides us a data-copying statistic with normalized expectation and variance under H_0. $Z_U \ll 0$ implies data-copying, $Z_U \gg 0$ implies underfitting. $Z_U < -5$ implies that if H_0 holds, Z_U is as likely as sampling a value < -5 from a standard normal. See Appendix 6.1 for consistency results.

This hypothesis test is a standard we can and should hold all generative models to. It is completely model agnostic and uses no estimate of likelihood. It only requires a meaningful distance metric, which is becoming common practice in the evaluation of mode-collapse and -dropping (Heusel et al., 2017; Sajjadi et al., 2018).

Additionally, the equal expectation condition of Equation 2 alone is almost sufficient to indicate a maximum likelihood Gaussian KDE model, like that in Equation 1, where the posterior probability that a random draw $x \sim q_\sigma(x)$ comes from the Gaussian component centered at training point t is

$$Q_\sigma(t|x) = \frac{\exp(-\|x-t\|^2/(2\sigma^2))}{\sum_{t' \in T} \exp(-\|x-t'\|^2/(2\sigma^2))}$$

Lemma 1. *For the kernel density estimator (1), the maximum-likehood choice of σ, namely the maximizer of $\mathbb{E}_{X\sim P}[\log q_\sigma(X)]$, satisfies*

$$\mathbb{E}_{X\sim P}\Big[\sum_{t\in T} Q_\sigma(t|X)\|X-t\|^2\Big] = \mathbb{E}_{Y\sim Q_\sigma}\Big[\sum_{t\in T} Q_\sigma(t|Y)\|Y-t\|^2\Big]$$

See Appendix 6.2 for proof. Unless σ is large, we know that for any given $x \in \mathcal{X}$, $\sum_{t\in T} Q_\sigma(t|x)\|x-t\|^2 \approx d(x) = \min_{t\in T}\|x-t\|^2$. So, enforcing that $\mathbb{E}_{X\sim P}[d(X)] = \mathbb{E}_{Y\sim Q}[d(Y)]$, and more loosely that $\mathbb{E}_{\substack{A\sim L(P)\\ B\sim L(Q)}}[\mathbb{1}_{B>A}] = \frac{1}{2}$ provides an excellent non-parametric approach to selecting a Gaussian KDE, and ought to be enforced for any Q attempting to emulate P.

2.3 Handling Local Heterogeneity

As described in Section 2.1, the above global test can be fooled by generators Q which are very close to the training data in some regions of the instance space (overfitting) but very far from the training data in others (poor modeling).

To handle heterogeneity in practice, we introduce *local* versions of our earlier tests. Let Π denote any partition of the instance space $\mathcal{X}$, which can be constructed in any manner. In our experiments, for instance, we run the k-means algorithm on T, so that $|\Pi| = k$. As the number of training and test samples grows, we may increase k and thus the instance-space resolution of our test. Letting $L_\pi(D) = L(D|_\pi)$ be the distribution of distances-to-training-set within cell $\pi \in \Pi$, we probe each cell of the partition individually.

Data Copying To offer a summary statistic for data copying, we collect the z-scored Mann-Whitney U statistic, Z_U, described in Section 2.2 in each cell π. Let $P_n(\pi) = |\{x : x \in P_n, x \in \pi\}|/n$ denote the fraction of P_n points lying in cell π, and similarly for $Q_m(\pi)$. The Z_U test for cell π and training set T will then be denoted as $Z_U\big(L_\pi(P_n), L_\pi(Q_m); T\big)$, where $L_\pi(P_n) = \{d(x) : x \in P_n, x \in \pi\}$ and similarly for $L_\pi(Q_m)$. See **Figure 1c** for examples of these in-cell scores. For stability, we only measure data-copying for those cells significantly represented by Q, as determined by a threshold τ. Let Π_τ be the set of all cells in the partition Π for which $Q_m(\pi) \geq \tau$. Then, our summary statistic for data copying averages across all cells represented by Q:

$$C_T(P_n, Q_m) := \frac{\sum_{\pi \in \Pi_\tau} P_n(\pi) Z_U\big(L_\pi(P_n), L_\pi(Q_m); T\big)}{\sum_{\pi \in \Pi_\tau} P_n(\pi)}$$

Representation The above test will not catch a model that heavily over- or under-represents cells. For completeness, we make use of a simple representation test that is essentially used by Richardson and Weiss (2018), now with an independent test set instead of the training set.

With $n, m \geq 20$ in cell π, we may treat $Q_m(\pi), P_n(\pi)$ as Gaussian random variables. We then check the null hypothesis $H_0 : 0 = P(\pi) - Q(\pi)$. Assuming this null hypothesis, a simple z-test is:

$$Z_\pi = \frac{Q_m(\pi) - P_n(\pi)}{\sqrt{\widehat{p}\big(1 - \widehat{p}\big)\big(\frac{1}{n} + \frac{1}{m}\big)}}$$

where $\widehat{p} = \frac{nP_n(\pi) + mQ_m(\pi)}{n+m}$. We then report two values for a significance level $s = 0.05$: the number of significantly different cells ('bins') with $Z_\pi > s$ (NDB over-representing), and the number with $Z_\pi < -s$ (NDB under-representing).

Together, these summary statistics — C_T, NDB-over, NDB-under — establish a tension that encourages Q to broadly represent P without directly copying training set T.

3 Experiments

Having clarified what we mean by data-copying in theory, we turn our attention to observing data copying by generative models in practice. We leave representation test results for the appendix, since this behavior has been well studied in previous works. Specifically, we aim to answer the two following questions:

1. Are the existing tests that measure generative model overfitting able to capture data-copying?
2. When popular generative models range from over- to underfit, does our test indicate data-copying, and if so, to what degree?

Methods In all of the following experiments, we select a training dataset T with test split P_n, and a generative model Q producing sample Q_m. We perform k-means on T to determine partition Π, with the objective having a reasonable population of both T and P_n in each $\pi \in \Pi$. We set threshold τ, such that we are guaranteed to have at least 20 samples in each cell in order to validate the gaussian assumption of Z_π, Z_U. We then select some parameter of Q that can tune the degree of over- or underfitting on training set T. For instance, the Gaussian KDE σ parameter will directly control the degree of data-copying by our definition, allowing us to sweep σ from low (complex, over-fit model) to high (simple, underfit model). For VAEs, we have no such parameter, and instead vary the model complexity from high (many units per layer) to low (few units per layer). We then probe for the degree of data-copying at each level of declining model complexity using the baseline and proposed methods, and record test responses. To observe the variance of each test, we record the average and 1-standard deviation of the test response across several trials of generating Q_m.

We embed all image samples into some latent space with meaningful L_2 distance to make $d(x)$ significant. While this is standard practice in evaluating generative models (Salimans et al., 2016; Sajjadi et al., 2018), these embeddings themselves might be overfit. To address this, we perform our experiments in three domains with three different kinds of embeddings (none, custom, and Inception Network Pool3 features).

3.1 Detecting data-copying

First, we investigate which of the existing generative model tests can detect explicit data-copying.

Baselines and Dataset Here, we probe the four of the methods described in our Related Work section, to see how they react to data-copying: two-sample NN (Lopez-Paz and Oquab, 2016), FID (Heusel et al., 2017), Binning-Based Evaluation (Richardson and Weiss, 2018), and Precision & Recall (Sajjadi et al., 2018), which are described in detail in Appendix 6.3.2. We run this test on the two-dimensional 'moons' dataset, as it affords us limitless training and test samples and requires no feature embedding (see Appendix 6.3.1 for an example). Note that, without an embedding, FID is simply the Frechét distance between two MLE normal distributions fit to T and Q_m. We use the same size generated and training sample for all methods, when $m < |T|$ (especially for large datasets and computationally burdensome samplers) we are forced to use an m-size training subsample $\widetilde{T}$ for running the two-sample NN test due to its constraint that $m = |T|$.

We compare against the canonical test of measuring the generalization gap (difference between training and test set likelihoods under the model) in Appendix 7. It is not a primary baseline since it cannot be used when the model likelihood is intractable.

We make Q a Gaussian KDE since it allows us to force explicit data-copying by setting σ very low. As $\sigma \to 0$, Q becomes a bootstrap sampler of the original training

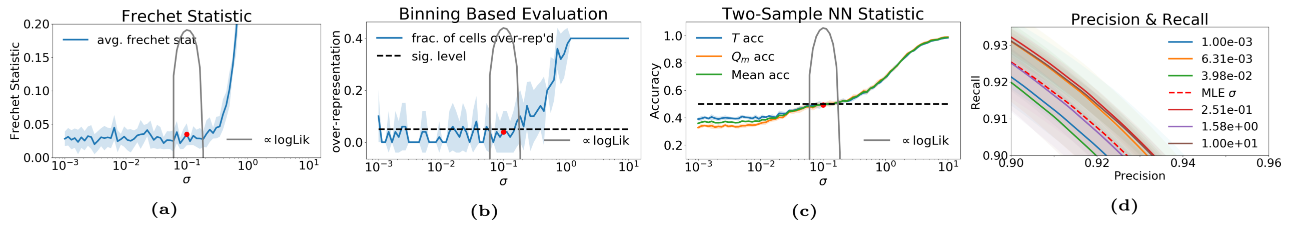


Figure 2: Response of four baseline test methods to data-copying of a Gaussian KDE on 'moons' dataset. Only the two-sample NN test **(c)** is able to detect data-copying KDE models as σ moves below σ_{MLE} (depicted as a red dot). The gray trace is proportional to the KDE's log-likelihood measured on a held-out validation set.

set. If a given test method can detect the level of data-copying by Q on T, it will provide a different response to a heavily over-fit KDE Q ($\sigma \ll \sigma_{\text{MLE}}$), a well-fit KDE Q ($\sigma \approx \sigma_{\text{MLE}}$), and an underfit KDE Q ($\sigma \gg \sigma_{\text{MLE}}$).

Figure 2 depicts how each baseline method responds to KDE Q models of varying degrees of data-copying, as Q ranges from data-copying ($\sigma = 0.001$) up to heavily underfit ($\sigma = 10$). The Frechét and Binning methods report effectively the same value for all $\sigma \leq \sigma_{\text{MLE}}$, indicating inability to detect data-copying. Similarly, the PR curves for different σ values are high variance and show no meaningful order with respect to σ.

The two-sample NN test does show a mild change in response as σ decreases below σ_{MLE}. This makes sense; as points in Q_m become closer to points in T, the two-sample NN accuracy should steadily drop to zero. The reason it does not drop to zero is due to the m subsampled training points, $\widetilde{T} \subset T$, needed to perform this test. As such, each training point $t \in T$ being copied by generated point $q \in Q_m$ is unlikely to be present in $\widetilde{T}$ during the test. This phenomenon is especially pronounced in some of the following settings.

The reason most of these tests fail to detect data-copying is because most existing methods focus on another type of overfitting: mode-collapse and -dropping, wherein entire modes of P are either forgotten or averaged together. However, if a model begins to data-copy, it is definitively overfitting *without* mode-collapsing.

Next, we will demonstrate our method on a variety of datasets, models, and embeddings. We will compare our method to the two-sample NN method in each setting, as it is the only baseline that responds to explicit data-copying.

3.2 Measuring degree of data-copying

We now aim to answer the second question raised at the beginning of this section: does $C_T(P_n, Q_m)$ detect and quantify data-copying? We focus on two types of generative model: Gaussian KDEs, and neural models.

3.2.1 KDE-based tests

While KDEs do not provide a reliable likelihood in high dimension (Theis et al., 2016), they do provide an informative first benchmark of the C_T statistic. KDEs allow us to directly force data-copying, and confirm the theoretical connection between the MLE KDE and $C_T \approx 0$ described in Lemma 1.

KDEs: 'moons' dataset Here, we repeat the experiment performed in Section 3.1, now including the C_T statistic for comparison. Refer to Appendix 6.3.1 for experimental details, and examples of the dataset.

Figures 3a and **3b** give a side-by-side depiction of C_T and the two-sample NN test accuracies across a range of KDE σ values. Think of C_T values as z-score standard deviations. We see that the C_T statistic in **Figure 3a** precisely identifies the MLE model when $C_T \approx 0$, and responds sharply to σ values above and below σ_{MLE}. The baseline in **Figure 3b** similarly identifies the MLE Q model when training accuracy ≈ 0.5, but is higher variance and less sensitive to changes in σ, especially for over-fit $\sigma \ll \sigma_{\text{MLE}}$. We will see in the next experiment, that this test breaks down for more complex datasets when $m \ll |T|$.

KDEs: MNIST Handwritten Digits We now extend the KDE test performed on the moons dataset to the significantly more complex MNIST handwritten digit dataset (LeCun and Cortes, 2010).

While it would be convenient to directly apply the KDE σ-sweeping tests discussed in the previous section, there are two primary barriers. The first is that KDE model relies on L_2 norms being perceptually meaningful, which is well understood not to be true in pixel space. The second problem is that of dimensionality: the 784-dimensional space of digits is far too high for a KDE to be even remotely efficient at interpolating the space.

To handle these issues, we first embed each image, $x \in \mathcal{X}$, to a perceptually meaningful 64-dimensional latent code, $z \in \mathcal{Z}$. We achieve this by training a convolutional autoencoder with a VGGnet perceptual

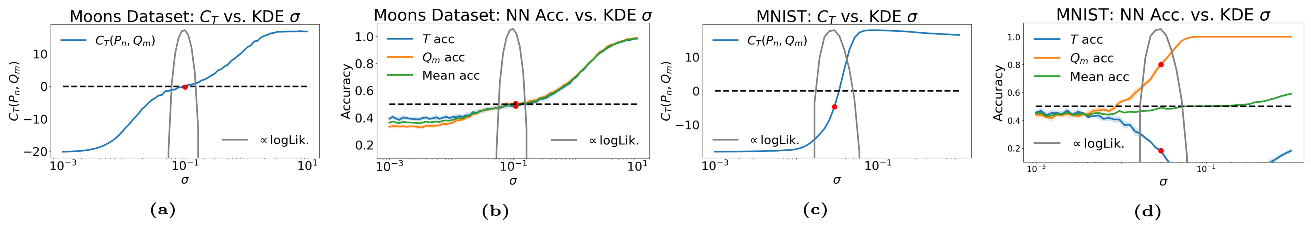


Figure 3: $C_T(P_n, Q_m)$ vs. NN baseline and generalization gap on moons and MNIST digits datasets. **(a,b)** compare the two tests on the moons dataset. **(c,d)** compare the two tests on MNIST. In both data settings, the C_T statistic is far more sensitive to the data-copying regime $\sigma \ll \sigma_{\text{MLE}}$ than the NN baseline. It is more sensitive to underfitting $\sigma \gg \sigma_{\text{MLE}}$ than the generalization gap test. The red dot denotes σ_{MLE}, and the gray trace is proportional to the KDE's log-likelihood measured on a held-out validation set.

loss produced by Zhang et al. (2018) (see Appendix 6.3.3 for more detail). Surely, even in the lower 64-dimensional space, the KDE will suffer some from the curse of dimensionality. We are not promoting this method as a powerful generative model, but rather as an instructive tool for probing a test's response to data-copying in the image domain.

Figure 3c shows how $C_T(P_n, Q_m)$ reacts decisively to over- and underfitting. It falsely determines the MLE σ value as slightly over-fit. However, the region of where C_T transitions from over- to underfit (say $-13 \leq C_T \leq 13$) is relatively tight and includes the MLE σ.

Meanwhile, **Figure 3d** shows how — with the generated sample smaller than the training sample, $m \ll |T|$ — the two-sample NN baseline provides no meaningful estimate of data-copying. In fact, the most data-copying models with low σ achieve the best scores closest to 0.5. Again, we are forced to use the m-subsampled $\widetilde{T} \subset T$, and most instances of data copying are completely missed. C_T has no such restriction.

These results are promising, and demonstrate the reliability of this hypothesis testing approach to probing for data-copying across different data domains. In the next section, we explore how these tests perform on more sophisticated, non-KDE models.

3.3 Neural Model Tests

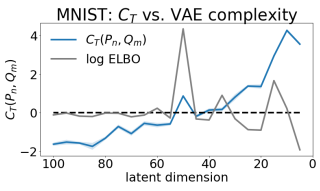
Gaussian KDE's may have nice theoretical properties, but are relatively ineffective in high-dimensional settings, precluding domains like images. As such, we also demonstrate our experiments on more practical neural models trained on higher dimensional image datasets (MNIST and ImageNet), with the goal of observing whether the C_T statistic indicates data-copying as these models range from over- to underfit.

MNIST VAE Here, we employ our data-copying test, $C_T(P_n, Q_m)$, on a range of VAEs of varying complexity trained on the MNIST handwritten digit dataset. Experimental and theoretical findings have suggested that VAE samplers — under certain assumptions — simply produce convex combinations of training set samples (Bozkurt et al., 2018). In generating an out-of-distribution sample, an overly complex VAE effectively reproduces nearest-neighbor training samples. Our findings appear to corroborate this. We vary model complexity by increasing the width (neurons per layer) in a three-layer VAE (see Appendix 6.3.3 for details). As an embedding, we pass all samples through the the convolutional autoencoder of Section 3.2.1, and collect statistics in this 64-dimensional space.

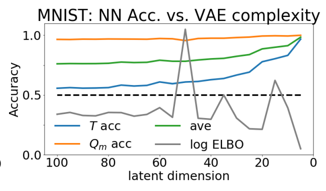
Figures 4a and **4b** compare the C_T statistic to the NN accuracy baseline . C_T behaves as it did in the previous sections: more complex models over-fit, forcing $C_T \ll 0$, and less complex models underfit forcing it $\gg 0$. We note that the range of C_T values is far less dramatic, which is to be expected since the KDEs were forced to explicitly data-copy. As likelihood is not available for VAEs, we compute each model's ELBO on a 10,000 sample held out validation set, and plot it in gray. We observe that the ELBO spikes for models with C_T near 0.

The NN baseline in **Figure 4b** is less interpretable, and fails to capture the overfitting trend as C_T does. While all three test accuracies still follow the upward-sloping trend of **Figure 3b**, they do not indicate where the highest validation set ELBO is. Furthermore, the NN accuracy statistics are shifted upward when compared to the results of the previous section: all NN accuracies are above 0.5 for all latent dimensions. This is problematic. A test statistic's absolute score ought to bear significance between very different data and model domains like KDEs and VAEs.

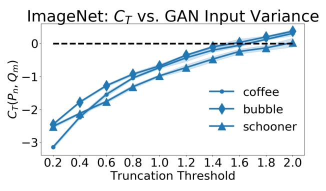
ImageNet GAN Finally, we scale our experiments up to a more practical image domain. We gather our test statistics on a state of the art conditional GAN, '*BigGan*' (Brock et al., 2018), trained on the Imagenet 12 dataset (Russakovsky et al., 2015). Conditioning on an input code, this GAN will generate one of 1000 different Imagenet classes. We run our experiments separately on three classes: 'coffee', 'soap bubble', and 'schooner'. All generated, test, and training images are



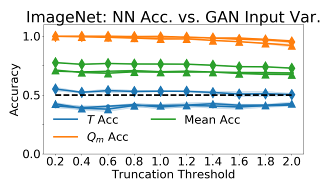
(a)



(b)

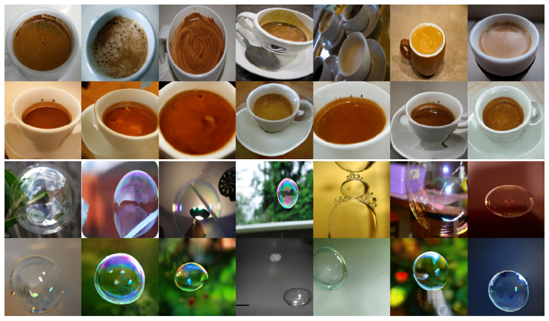


(c)

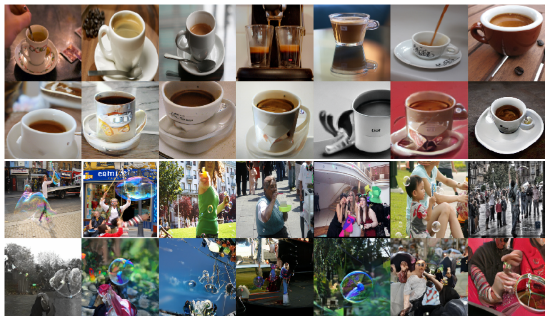


(d)

Figure 4: Neural model data-copying: figures **(a)** and **(c)** demonstrate the C_T statistic identifying data-copying in an MNIST VAE and ImageNet GAN as they range from heavily over-fit to underfit. **(b)** and **(d)** demonstrate the relative insensitivity of the NN baseline to this overfitting. (Note, the markers for **(d)** apply to the traces of **(e)**)



(a) Data-copied cells; top: $Z_U = -1.46$, bottom: $Z_U = -1.00$



(b) Underfit cells; top: $Z_U = +1.40$, bottom: $Z_U = +0.71$

Figure 5: Data-copied and underfit cells of ImageNet12 'coffee' and 'soap bubble' instance spaces (trunc. threshold = 2). In each 14-figure strip, the top row provides a random series of training samples from the cell, and the bottom row provides a random series of generated samples from the cell. **(a)** Data-copied cells. **(a), top**: Random training and generated samples from a $Z_U = -1.46$ cell of the coffee instance space. **(a), bottom**: Random training and generated samples from a $Z_U = -1.00$ cell of the bubble instance space. **(b)** Underfit cells. **(b), top**: Random training and generated samples from a $Z_U = +1.40$ cell of the coffee instance space. **(b), bottom**: Random training and generated samples from a $Z_U = +0.71$ cell of the bubble instance space.

embedded to a 64-dimensional space by first gathering the 2048-dimensional features of an InceptionV3 network 'Pool3' layer, and then projecting them onto the 64 principal components of the training embeddings. Appendix 6.3.4 has more details.

Being limited to one pre-trained model, we increase model variance ('truncation threshold') instead of decreasing model complexity. As proposed by *BigGan*'s authors, all standard normal input samples outside of this truncation threshold are resampled. The authors suggest that lower truncation thresholds, by only producing samples at the mode of the input, output higher quality samples at the cost of variety, as determined by Inception Score (IS). Similarly, the FID score finds suitable variety until truncation approaches zero.

As depicted in **Figure 4c**, the C_T statistic remains well below zero until the truncation threshold is nearly maximized, indicating that Q produces samples closer to the training set than real samples tend to be. While FID finds that *in aggregate* the distributions are roughly similar, a closer look suggests that Q allocates too much probability mass near the training samples.

Meanwhile, the two-sample NN baseline in **Figure 4d** hardly reacts to changes in truncation, even though the generated and training sets are the same size, $m = |T|$. Across all truncation values, the training sample NN accuracy remains around 0.5, not quite implying over- or underfitting.

A useful feature of the C_T statistic is that one can examine the Z_U scores it is composed of to see which of the cells $\pi \in \Pi_\tau$ are or are not copying. **Figure 5** shows the samples of over- and underfit cells $\pi \in \Pi$ for two of the three classes. For both 'coffee' and 'bubble' classes, the underfit cells are more diverse than the data-copied cells. While it might seem reasonable that these generated samples are further from nearest neighbors in more diverse clusters, keep in mind that the $Z_U > 0$ statistic indicates that they are further from training neighbors than test set samples are. For instance, the people depicted in underfit 'bubbles' cell are highly distorted.

4 Conclusion

In this work, we have formalized *data-copying*: an under-explored failure mode of generative model overfitting. We have provided preliminary tests for measuring data-copying and experiments indicating its presence in a broad class of generative models. In future work, we plan to establish more theoretical properties of data-copying, convergence guarantees of these tests, and experiments with different model parameters.

5 Acknowledgements

We thank Rich Zemel for pointing us to Xu et al. (2018), which was the starting point of this work. Thanks to Arthur Gretton and Ruslan Salakhutdinov for pointers to prior work, and Philip Isola and Christian Szegedy for helpful advice. Finally, KC and CM would like to thank N00014-16-1-261, UC Lab Fees under LFR 18-548554 and NSF IIS 1617157 for research support.

References

Wacha Bounliphone, Eugene Belilovsky, Matthew B. Blaschko, Ioannis Antonoglou, and Arthur Gretton. A test of relative similarity for model selection in generative models. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.04581`.

Alican Bozkurt, Babak Esmaeili, Dana H Brooks, Jennifer G Dy, and Jan-Willem van de Meent. Can vaes generate novel examples? 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *CoRR*, abs/1612.02136, 2016. URL `http://arxiv.org/abs/1612.02136`.

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

Ishaan Gulrajani, Colin Raffel, and Luke Metz. Towards GAN benchmarks which require generalization. *CoRR*, abs/2001.03653, 2020. URL `https://arxiv.org/abs/2001.03653`.

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. 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. Advances in Neural Information Processing Systems 33 (NIPS 2019), 2019.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

Eitan Richardson and Yair Weiss. On gans and gmms. In *Advances in Neural Information Processing Systems*, pages 5847–5858, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. Advances in Neural Information Processing Systems 31 (NIPS 2017), 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

Dougal J. Sutherland, Hsiao-Yu Fish Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alexander J. Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. *CoRR*, abs/1611.04488, 2016. URL `http://arxiv.org/abs/1611.04488`.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.01844`.

Ryan Webster, Julien Rabin, Loïc Simon, and Frédéric Jurie. Detecting overfitting of deep generative networks via latent recovery. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11273–11282. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.01153. URL `http://openaccess.thecvf.com/content_CVPR_2019/html/Webster_Detecting_Overfitting_of_Deep_Generative_Networks_via_Latent_Recovery_CVPR_2019_paper.html`.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger B. Grosse. On the quantitative analysis of decoder-based generative models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL `https://openreview.net/forum?id=B1M8JF9xx`.

Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.