Yale University CPSC 683
Stability in Machine Learning: Generalization, Privacy & Replicability

Lecture 9: Memorization in Machine Learning

Instructor: Alkis Kalavasis, alkis.kalavasis@yale.edu

Lecture 9 focuses on two important theoretical works by Feldman (2020) and Brown et al. (2021) on memorization in Machine Learning, and its connections to privacy and stability. Moreover, we discuss memorization in generative models and, in particular, diffusion models (Shah et al., 2025).

1 Memorization, Learning, and Privacy

Deep learning models are over-parameterized, meaning they have far more tunable parameters than available data points. As a result, they can easily "overfit" by memorizing training labels rather than using the training points to generalize to unseen data. As we have already seen in prior lectures, state-of-the-art image recognition models, which achieve near-perfect training accuracy, are known to fit even randomly assigned labels (Zhang et al., 2021). The main question we would like to understand is the following:

Is memorization necessary for learning (e.g., image classification) and generative modeling (e.g., diffusion models)?

Do near-optimal classifiers have to memorize training examples? Apart from classification, questions about generation and memorization have deep connections to various areas of research such as linguistics (Slobin, 2013).

Beyond its theoretical implications, memorization raises serious *privacy risks*, particularly when training data includes sensitive personal information.

- Regarding classification, there are various works studying memorization or overfitting (Zhang et al., 2019, 2021; Belkin et al., 2018). Many works focus on *label memorization* e.g., Feldman (2020); Arpit et al. (2017); Ma et al. (2018) or on memorization of the entire training set (Brown et al., 2021). Zhang et al. (2021) empirically demonstrate that DNNs are capable of fitting random data, which implicitly necessitates some high degree of memorization. Feldman (2020) rigorously shows that, for some problems, label memorization is necessary for achieving near-optimal accuracy on test data.
- Carlini et al. (2023); Daras et al. (2023, 2024); Somepalli et al. (2022, 2023); Ross et al. (2024) show that diffusion models memorize the training data and often replicate them at generation time.
- Carlini et al. (2021) demonstrate that modern models for next-word prediction memorize large chunks of text from the training data verbatim, including personally identifiable and sensitive information such as phone numbers.

2 Regularization and Memorization

The standard theoretical framework for preventing overfitting relies on *regularization*, which ensures a balance between model complexity and empirical error. The idea is that fitting outliers requires increasing model complexity (intuitively increasing the VC dimension or model capacity), and by tuning this trade-off, algorithms can capture meaningful patterns without overfitting. However, this perspective contradicts empirical observations in modern deep learning.

- Models trained on image and text classification datasets often achieve near-perfect (95–100%) accuracy on training data.

- This happens even when test accuracy is significantly lower (typically 50–80%).
- Such high training accuracy (with low test error) implies memorization of mislabeled data and outliers, which are unavoidable in large datasets.
- Moreover, even with strong regularization, models achieve over 90% training accuracy on ImageNet even when labels are randomly assigned (Zhang et al., 2021). This suggests that current regularization techniques are not strong enough to prevent memorization.

3 Label Memorization: A Short Tale about a Long Tail

Feldman (2020) proposes a conceptually simple explanation and supporting theory for

why memorization of seemingly useless labels may be necessary to achieve close-to-optimal generalization error.

It is based on the viewpoint that the primary hurdle to learning an accurate model is not the noise inherent in the labels but rather an insufficient amount of data to predict accurately on rare and atypical instances.

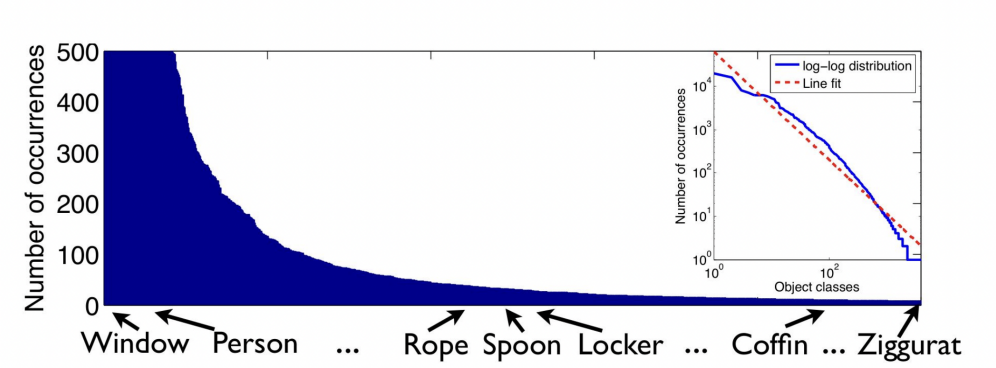
Insight 1: TLDR

Feldman (2020) theoretically shows strong trade-offs between memorization and generalization by showing that memorization is *necessary* for (optimal) *classification*.
The need for memorization in (Feldman, 2020) is associated with the frequencies of different subpopulations (e.g., cats, dogs, etc.) that appear in the dataset. The key observation is that the distribution of the frequencies is usually *heavy-tailed* (Zhu et al., 2014), i.e., roughly speaking in a dataset of size n, there will be many classes with frequency around $1/n$. This means that the training algorithm will only observe a single representative from those subpopulations and cannot distinguish between the following two cases:
Case 1. If the unique example comes from an extremely rare subpopulation (with frequency $\ll 1/n$), then memorizing it has no significant benefits, and,
Case 2. If the unique example comes from a subpopulation with $1/n$ frequency, then memorizing it will probably improve the accuracy on the entire subpopulation and decrease the generalization error by $\Omega(1/n)$. Hence, the optimal classifier should memorize these unique examples to avoid paying Case 2 in the error.
Rare or atypical instances are also the cause of language model hallucinations. The existence of such rare instances was the key tool for Kalai and Vempala to prove that calibrated language models must hallucinate (Kalai and Vempala, 2024).

3.1 What is a "rare" example?

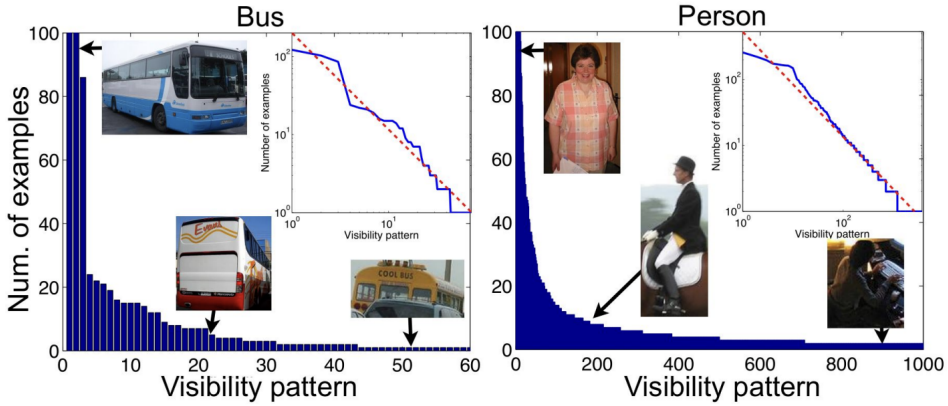
In practice, rare instances are referred as the "long tail" of the data distribution. Modern datasets used for visual object recognition and text labeling follow classical long-tailed distributions such as Zipf distribution (or more general power law distributions); i.e., the frequencies of the classes of the datasets follow a heavy-tailed distribution (see Figure 1).

To be more specific, let us consider a multiclass prediction problem. To formalize the notion of having a "long (or heavy) tail", we will model the data distribution of each class *as a mixture of distinct subpopulations*. For example, images of birds include numerous different species photographed from different perspectives and under different conditions (such as close-ups, in foliage and in the sky). Naturally, the subpopulations may have different frequencies (which correspond to mixture coefficients). We model long-tailed data distributions as distributions in which the frequencies of subpopulations are long-tailed (see Figure 2).



(a) The number of examples by object class in SUN dataset

Figure 1: Taken from Feldman (2020). The distribution of the frequencies of the classes in a dataset is heavy-tailed.



(b) Distributions of the visibility patterns for bus and person

Figure 2: Taken from Feldman (2020). For a fixed (coarse) label (e.g., bus or person), the distribution of the frequencies of the "sub-populations" (fine labels) in a dataset is heavy-tailed.

3.2 What do we predict on "rare" examples?

We make two reasonable hypotheses:

1. Before seeing the dataset, the learning algorithm does not know the frequencies of subpopulations.
2. The algorithm is not able to predict accurately on a subpopulation until at least one example from the subpopulations is observed.

Feldman's reasoning then goes as follows:

1. A dataset of n samples from a long-tailed mixture distribution will have some subpopulations from which just a single example was observed (and some subpopulations from which none at all).
2. To predict more accurately on a subpopulation from which only a single example was observed (and to fit the example) the learning algorithm needs to memorize the label of the example.

Is this behavior necessary for close-to-optimal generalization error. The crucial observation of Feldman (2020) is that this depends on the frequency of the subpopulation.

- If the unique example from a subpopulation (or singleton) comes from an extremely rare (or "outlier") subpopulation (say with mixing coefficient 2^{-n}) then memorizing it has no significant benefits.
- At the same time, if the singleton comes from an "atypical" subpopulation with frequency on the order of $1/n$, then memorizing such an example is likely to improve the accuracy on the entire subpopulation and thereby reduce the generalization error by $\Omega(1/n)$.

The key point of Feldman (2020) is that based on observing a single sample from a subpopulation, *it is impossible to distinguish samples from "atypical" subpopulations from those in the "outlier" ones.*

Hence, an algorithm can only avoid the risk of missing "atypical" subpopulations by memorizing the labels of singletons from the "outlier" subpopulations (since it cannot distinguish between them). The next step is to measure the mass of the $1/n$ coefficients: in a long-tailed distribution of frequencies, the total weight of frequencies on the order of $1/n$ is significant enough that ignoring these subpopulations will hurt the generalization error substantially.

Combining the above, any algorithm that achieves near-optimal generalization against such heavy-tailed instances, has to memorize the labels of outliers.

3.3 The Key Result

Consider a discrete population $X = [N]$ and label space $Y = [m]$.

Consider a set of functions $\mathcal{F}$ mapping X to Y. Fix a known prior $\pi = (\pi_1, ..., \pi_N)$. We consider a random distribution over X as follows: for any $x \in X$, set p_x randomly and independently from π. Define the pdf on X as $D(x) = p_x / \sum_x p_x$. We also let $\overline{\pi}$ the resulting marginal distribution over the frequency of any single element in x.

An instance of our learning problem is generated by picking a random $D \sim \mathcal{D}_\pi$ and a random true labeling $f \sim \mathcal{F}$.

We let

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}) = \mathop{\mathbf{E}}_{D \sim \mathcal{D}_\pi, f \sim \mathcal{F}} \mathrm{err}_{D,f}(\mathcal{A}) .$$

We can now define

$$\tau_1 = \frac{\mathbf{E}_a[a^2(1-a)^{n-1}]}{\mathbf{E}_a[a(1-a)^{n-1}]}$$

where a is drawn from the actual marginal distribution over frequencies that results from our process (see Feldman (2020)). This essentially captures the expected frequency of a sample conditioned on observing it in the dataset exactly once. We can extend this definition to τ_ℓ for $\ell \in [n]$.

The main result is as follows.

Theorem 1

For every learning algorithm $\mathcal{A}$ and any dataset $Z \in (X \times Y)^n$,

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}|Z) \geq \mathrm{opt}(\pi, \mathcal{F}|Z) + \sum_{\ell \in [n]} \tau_\ell \cdot \mathrm{errn}_Z(\mathcal{A}, \ell) .$$

Proof. The key observation is that D and f are picked independently during the generation of the learning instance. We write $G(\cdot|Z)$ as the distribution that generates the pair (D, f) conditioned on Z; note that this is a product measure. We can write

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}|Z) = \mathop{\mathbf{E}}_{(D,f) \sim G(\cdot|Z)} \mathop{\mathbf{E}}_{h \sim \mathcal{A}(Z)} \sum_{x \in X} 1\{h(x) \neq f(x)\} D(x) .$$

We now decompose X as the elements x that do not appear in the training set Z (i.e., $x \in X_{Z\#0}$) and the elements x that appear exactly ℓ times ($x \in X_{Z\#\ell}$ for any $\ell \in [n]$). Let $X_Z = \cup_\ell X_{Z\#\ell}$. We can hence decompose $\sum_{x\in X} = \sum_{x\in X_{Z\#0}} + \sum_{x\in X_Z}$. For every $x \in X_{Z\#0}$, we get

$$\mathop{\mathbf{E}}_{(D,f)\sim G(\cdot|Z)} \mathop{\mathbf{E}}_{h\sim\mathcal{A}(Z)} 1\{h(x) \neq f(x)\}D(x) = \mathop{\mathbf{Pr}}_{f\sim\mathcal{F},h\sim\mathcal{A}(Z)}[h(x) \neq f(x)] \mathop{\mathbf{E}}_{D\sim\mathcal{D}(\cdot|Z)}[D(x)]\,.$$

Similarly, for $x \in X_{Z\#\ell}$,

$$\mathop{\mathbf{E}}_{(D,f)\sim G(\cdot|Z)} \mathop{\mathbf{E}}_{h\sim\mathcal{A}(Z)} 1\{h(x) \neq f(x)\}D(x) = \mathop{\mathbf{Pr}}_{f\sim\mathcal{F},h\sim\mathcal{A}(Z)}[h(x) \neq f(x)] \mathop{\mathbf{E}}_{D\sim\mathcal{D}(\cdot|Z)}[D(x)]\,,$$

and we observe that for that x (which appears exactly ℓ times):

$$\mathop{\mathbf{E}}_{D\sim\mathcal{D}(\cdot|Z)}[D(x)] = \tau_\ell\,.$$

Hence, we get that

$$\overline{\mathrm{err}}(\pi,\mathcal{F},\mathcal{A}|Z) = \sum_{\ell\in[n]} \tau_\ell \cdot \mathrm{errn}_Z(\mathcal{A},\ell) + \sum_{x\in X_{Z\#0}} \mathop{\mathbf{Pr}}_{f,h}[h(x) \neq f(x)] \mathop{\mathbf{E}}_{D}[D(x)]\,.$$

This implies the claimed inequality since the right-hand side is minimizes when for all $\ell \in [n]$, $\mathrm{errn}_Z(\mathcal{A},\ell) = 0$ and for all unseen x, it holds that $\mathbf{Pr}_{h,f}[h(x) \neq f(x)] = \min_{y\in Y} \mathbf{Pr}_f[f(x) \neq y]$. Note that this minimum is minimized by the algorithm $\mathcal{A}^*$ that memorizes the examples in Z and predicts the minimizing label y for all $x \in X_{Z\#0}$. Hence, $\overline{\mathrm{err}}(\pi,\mathcal{F},\mathcal{A}^*|Z) = \mathrm{opt}(\pi,\mathcal{F})$.

□

We can remove the conditioning on Z by taking an expectation over the Z-marginal.

Note that the above can be generalized to continuous populations. Here, we model the unlabeled data distribution as a mixture of a large number of fixed distributions $M_1, ..., M_N$ and we sample the mixing coefficients using the same random process as before to get a mixture $M(x) = \sum_i D_i M_i(x)$. To put some structure, we assume that the entire subpopulation X_i is labeled by the same label and now we look at the multiplicity of sub-domains and not points themselves and count mistakes just once per sub-domain.

3.4 Heavy-Tailed Distributions

We are going to formally explain what it means for the frequencies of the original dataset to be heavy-tailed (Zhu et al., 2014; Feldman, 2020). This heavy-tailed structure will then allow us to control the generalization error. We will be interested in subpopulations that have only one representative in the training set Z (these are the examples that will cost roughly τ_1 in the error). We will refer to them as *single* subpopulations.

For this to happen given that $|Z| = n$, it should be roughly speaking the case where some frequencies D_i are of order $1/n$. The quantity that controls how many of the frequencies D_i will be of order $1/n$ is the mass that the distribution $\overline{\pi}(a) = \mathbf{Pr}_D[D_i = a]$ assigns to the interval $[1/(2n), 1/n]$.

Typically, we will call a list of frequencies π *heavy-tailed* if

$$\mathrm{weight}\left(\overline{\pi}, \left[\frac{1}{2n}, 1/n\right]\right) = \Omega(1)\,. \tag{1}$$

In words, there should be a constant number of subpopulations with frequencies of order $O(1/n)$. This definition is important because it can then lower bound the value τ_1 and hence it can lower bound the generalization loss of not fitting single subpopulations.

Lemma 1

Consider a dataset of size n and assume that π is heavy-tailed, as in (1). Then $\tau_1 = \Omega(1/n)$.

On the contrary, when π is not heavy-tailed, τ_1 will be small and hence generalization is not hurt by not memorizing.

3.5 Memorization and Stability

Using the above key result, we have that

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}) \geq \mathrm{opt}(\pi, \mathcal{F}) + \tau_1 \cdot \mathop{\mathbf{E}}_{Z} \mathrm{errn}_Z(\mathcal{A}, 1)\,.$$

This inequality relates the sub-optimality gap of an arbitrary algorithm A to its error in elements that appear exactly once in the dataset.

Let us now introduce the notion of memorization. Let us, in particular, see the definition of *label memorization* according to Feldman (2020):

$$\mathrm{mem}(A, S, i) = \mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) = y_i] - \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) = y_i]$$

or

$$\mathrm{mem}(A, S, i) = \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) \neq y_i] - \mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) \neq y_i]\,.$$

We think of a label as memorized when this value is larger than some fixed positive constant.

Observe that this definition is closely related to the classical leave-one-out notion of stability but focuses on the change in the label and not in the incurred loss.

The expectation over the choice of dataset of the average memorization value is equal to the expectation of the generalization gap:

$$\frac{1}{n} \mathop{\mathbf{E}}_{S \sim P^n} \sum_{i \in [n]} \mathrm{mem}(A, S, i) = \mathop{\mathbf{E}}_{S' \sim P^{n-1}}[\mathrm{err}_P(A(S'))] - \mathop{\mathbf{E}}_{S \sim P^n}[\mathrm{err}_S(A(S))]$$

where

$$\mathrm{err}_S(A(S)) = \frac{1}{n} \sum_i \mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) \neq y_i]\,, \quad \mathrm{err}_P(A(S')) = \mathop{\mathbf{Pr}}_{(x,y) \sim P} \mathop{\mathbf{Pr}}_{h \sim A(S'))}[h(x) \neq y]\,.$$

Thus a large generalization gap implies that a significant fraction of labels is memorized. An immediate corollary of this definition is that an algorithm with a limited ability to memorize labels will not fit the singleton data points well whenever the algorithm cannot predict their labels based on the rest of the dataset.

Lemma 2

For any dataset $S \in (X \times Y)^n$, any algorithm A and $i \in [n]$,

$$\mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) \neq y_i] = \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) \neq y_i] - \mathrm{mem}(A, S, i)\,.$$

This means that

$$\mathrm{errn}_S(A, 1) = \sum_{i \in [n], x_i \in X_{S\#1}} \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) \neq y_i] - \mathrm{mem}(A, S, i)\,.$$

Hence, if the first term in the RHS is large (the algorithm cannot predict the label based on the rest of the training set) then memorization must be large in order to make the LHS small.

4 Memorization of Training Data

We next shortly discuss the results of the work of Brown et al. (2021). We will study instances in which most of the information about entire high-dimensional training examples (and not only the labels) must be encoded by near-optimal learning algorithms.

We define a problem instance P as a distribution over labeled examples, i.e., P is a distribution over where $\mathcal{S} = X \times Y$ is a space of examples X paired with labels in Y. A dataset $S \in \mathcal{S}^n$ is generated by sampling i.i.d. from such a distribution. We assume that d is the dimension of the data; hence S can be described by $\Theta(nd)$ bits (this will be important for the information-theoretic result that follows). The instance P is itself drawn from a meta-distribution q, dubbed the learning task. The learning task q is assumed to be known to the learner, but the specific problem instance is a priori unknown.

Given $S \sim P^n$, the algorithm A produces a model $M = A(S)$. We are interested in the following loss:

$$\mathrm{err}_{q,n}(A) = \mathop{\mathbf{Pr}}_{P\sim q} \mathop{\mathbf{Pr}}_{S\sim P^n} \mathop{\mathbf{Pr}}_{(z,y)\sim P} \mathop{\mathbf{Pr}}_{\text{coins}A,M} [M(z) \neq y]$$

Theorem 2

Assume that the examples lie in $\{0,1\}^d$. For all n and d, there exist natural tasks q for which any algorithm A that satisfies, for small constant ϵ,

$$\mathrm{err}_{q,n}(A) \leq \mathrm{err}_{q,n}(A_{OPT}) + \epsilon$$

also satisfies

$$I(S; M|P) = \Omega(nd)\,.$$

This essentially means that any algorithm with near-optimal accuracy on q must memorize $\Omega(nd)$ bits about the sample.

Insight 2: Interpretation of this result

Recall that the conditional mutual information is defined via two conditional entropy terms:

$$I(S; M|P) = H(S|P) - H(S|M, P)\,.$$

Consider an observer who knows the full data distribution P (but not X).

1. The term $I(S; M|P)$ captures how the observer's uncertainty about the set S is reduced after observing the model M.
2. The term $H(S|P)$ captures the uncertainty about what is unique to the data set, such as noise or irrelevant features.
3. So $I(S; M|P) = \Omega(nd)$ means that (i) the learning algorithm must encode a constant fraction of the information it receives (since after seeing M, there is a very large drop in the uncertainty), but also (ii) a constant fraction of what the model encodes is irrelevant to the task (since the drop is against $H(S|P)$ which are the unique properties of the dataset unrelated to the general task P).

Insight 3: Why do we need a meta-distribution?

The meta-distribution q captures the learner's initial uncertainty about the problem, and is essential to the result: if the exact distribution P were known to the learner A, it could simply ignore X and

write down an optimal classifier for P as its model M. In that case, we would have $I(X; M|P) = 0$. That said, since conditional information is an average over realizations of P, our result also means that for every learner, $I(M; X)$ is large for some particular p in the support of q.

5 Memorization and Diffusion Models

For a more empirical perspective on memorization and diffusion models, we refer to Somepalli et al. (2022, 2023); Daras et al. (2023); Shah et al. (2025).

In this section, we will shortly discuss about the work of Shah et al. (2025). We first mention how one measures empirically quality and memorization.

Quality We compute the Fréchet Inception Distance (Heusel et al., 2017) (FID) between 50,000 generated samples and 50,000 dataset samples as a measure of quality.

Memorization We measure memorization by computing the similarity score (i.e., inner product) of each generated sample to its nearest neighbor in the embedding space of DINOv2 (Oquab et al., 2023).

Diffusion The first step in diffusion modeling is to design a corruption process. We define a sequence of increasing corruption levels indexed by $t \in [0, 1]$, with:

$$X_t = \sqrt{1 - \sigma_t^2} X_0 + \sigma_t Z, \quad Z \sim \mathcal{N}(0, I_d), \tag{2}$$

where the map $\sigma_t := \sigma(t)$ is the noise schedule and X_0 is drawn from the clean distribution p_0. Our ultimate goal is to sample from the unknown distribution p_0. The key idea behind diffusion modeling is to learn the score functions, defined as $\nabla \log p_t(\cdot)$, for different noise levels t, where $X_t \sim p_t$.

one can train directly for the score function using the noise prediction loss (Ho et al., 2020; Vincent, 2011):

$$J(\theta) = \mathop{\mathbf{E}}_{x_0, x_t, t} \left[\left\| s_\theta(x_t, t) - \frac{\sqrt{1 - \sigma_t^2} x_0 - x_t}{\sigma_t^2} \right\|^2 \right]. \tag{3}$$

Given access to the score function for different times t, one can sample from the distribution of p_0 by running the process (Song et al., 2020):

$$\mathrm{d}x = \left(-x - \frac{(\mathrm{d}\sigma_t / \mathrm{d}t)\sigma_t}{1 + \sigma_t^2} \nabla \log p_t(x_t) \right) \mathrm{d}t. \tag{4}$$

The underlying distribution of x_0 is continuous, but in practice we only optimize this objective over a finite distribution of training points. Prior work has shown that when the expectation is taken over an empirical distribution $\widehat{p}_0$, the optimal score can be written in closed form (Scarvelis et al., 2023; Biroli et al., 2024; De Bortoli, 2022; Kamb and Ganguli, 2024; Benton et al., 2024). Specifically, the optimal score for the empirical distribution is a gradient filed where each point x_0 in the finite sample S (i.e., the empirical distribution $\widehat{p}_0$) is pulling the noisy iterate x_t towards itself, where the weight of the pull depends on the distance of each training point to the noisy point. The optimal score will lead to a diffusion model that only *replicates* the training points during sampling (Scarvelis et al., 2023; Kamb and Ganguli, 2024). Hence, any potential creativity that is observed experimentally in diffusion models comes from the failure to perfectly optimize the training objective.

Ambient Score Matching One way to mitigate memorization is to never see the training data. Consider the case where we are given samples from a noisy distribution p_{t_n} (where t_n stands for t-nature) and we desire to learn the score at time t for $t > t_n$. The Ambient Score Matching loss (Daras et al., 2024), defined as:

$$J_{\text{ambient}}(\theta) = \mathbb{E}_{x_{t_n}} \mathbb{E}_{(x_t,t)|x_{t_n}} \left[\left\| \frac{\sigma_t^2 - \sigma_{t_n}^2}{\sigma_t^2 \sqrt{1-\sigma_{t_n}^2}} h_\theta(x_t, t) + \frac{\sigma_{t_n}^2}{\sigma_t^2} \sqrt{\frac{1-\sigma_t^2}{1-\sigma_{t_n}^2}} x_t - x_{t_n} \right\|^2 \right], \tag{5}$$

can learn the conditional expectation $\mathbf{E}[x_0|x_t]$ without ever looking at clean data from p_0.

> **Exercise 1**
>
> Prove that Ambient Score matching learns the correct conditional expectation.

The intuition behind this objective is that to denoise the noisy sample x_t, we need to find the direction of the noise and then rescale it appropriately. The former can be found by denoising to an intermediate level t_n and the rescaling ensures that we denoise all the way to the level of clean images. Once the conditional expectation $\mathbf{E}[x_0|x_t]$ is recovered, we get the score by using Tweedie's Formula. We remark that this objective can only be used for $t > t_n$.

We are now ready to present our framework for training diffusion models with limited data that will allow creativity without sacrificing quality. Our key observation is that the diversity of the generated images is controlled in the high-noise part of the diffusion trajectory (Dieleman, 2024; Li and Chen, 2024). Hence, if we can avoid memorization in this regime, it is highly unlikely that we will replicate training examples at inference time, even if we memorize at the low-noise part. Our training algorithm can "copy" details from the training samples and still produce diverse outputs.

> **Insight 4**
>
> The key property of Feldman (2020) seems to break when noise is added to the images. That is because different subpopulations start to merge and the heavy-tails of the weights' distribution disappear. Interestingly, diffusion models learn the (score of the) distribution at different levels of noise. This indicates that, in principle, it is feasible to avoid memorization in the high-noise regime (without sacrificing too much quality).

Algorithm The algorithm of Shah et al. (2025) works by splitting the diffusion training time into two parts, $t \leq t_n$ and $t > t_n$, where t_n (t-nature) is a free parameter to be controlled. For the regime, $t \leq t_n$, we train with the regular diffusion training objective, and (assuming perfect optimization) we know the exact score. To train for $t > t_n$, we first create the set S_{t_n} which has *one* noisy version of each image in the training set. Then, we train using the set S_{t_n} and the Ambient Score Matching loss.

> **Insight 5**
>
> It is useful to build some intuition about why this algorithm avoids memorization and at the same time produces high-quality outputs. Regarding memorization: 1) the learned score function for times $t \geq t_n$ does not point directly towards the training points since Ambient Diffusion aims to predict the noisy points (recall that the optimal DDPM solution points towards scalings of the training points) and 2) the noisy versions x_{t_n} are harder to memorize than x_0, since noise is not compressible. At the same time, if the dataset size were to grow to infinity, both our algorithm and the standard diffusion objective would find the true solution: the score of the underlying continuous distribution. In fact,

Algorithm 1 Algorithm for training diffusion models using limited data.

Require: untrained network h_θ, set of samples S, noise level t_n, noise scheduling $\sigma(t)$, batch size B, diffusion time T

$S_{t_\text{n}} \leftarrow \{\sqrt{1-\sigma_{t_\text{n}}^2}\, x_0^{(i)} + \sigma_{t_\text{n}} \varepsilon^{(i)} | x_0^{(i)} \in S, \varepsilon^{(i)} \sim \mathcal{N}(0, I_d)\}$ ▷ Noise the training set at level t_n.
while not converged **do**
  Form a batch $\mathcal{B}$ of size B uniformly sampled from $S \cup S_{t_\text{n}}$
  loss $\leftarrow 0$ ▷ Initialize loss.
  for each sample $x \in \mathcal{B}$ **do**
    $\varepsilon \sim \mathcal{N}(0, I)$ ▷ Sample noise.
    if $x \in S_{t_\text{n}}$ **then**
      $x_{t_\text{n}} \leftarrow x$ ▷ We are dealing with a noisy sample.
      $t \sim \mathcal{U}(t_\text{n}, T)$ ▷ Sample diffusion time for noisy sample.
      $x_t \leftarrow \sqrt{\frac{1-\sigma_t^2}{1-\sigma_{t_\text{n}}^2}}\, x_{t_\text{n}} + \sqrt{\frac{\sigma_t^2-\sigma_{t_\text{n}}^2}{1-\sigma_{t_\text{n}}^2}}\, \varepsilon$ ▷ Add additional noise.
      loss $\leftarrow$ loss $+ \left\| \frac{\sigma_t^2-\sigma_{t_\text{n}}^2}{\sigma_t^2\sqrt{1-\sigma_{t_\text{n}}^2}} h_\theta(x_t, t) + \frac{\sigma_{t_\text{n}}^2}{\sigma_t^2}\sqrt{\frac{1-\sigma_t^2}{1-\sigma_{t_\text{n}}^2}}\, x_t - x_{t_\text{n}} \right\|^2$ ▷ Ambient Score Matching.
    else
      $x_0 \leftarrow x$ ▷ We are dealing with a clean sample.
      $t \sim \mathcal{U}(0, t_\text{n})$ ▷ Sample diffusion time for clean sample.
      $x_t \leftarrow \sqrt{1-\sigma_t^2}\, x_0 + \sigma_t \varepsilon$ ▷ Add noise.
      loss $\leftarrow$ loss $+ \|h_\theta(x_t, t) - x_0\|^2$ ▷ Regular Denoising Score Matching.
    end if
  end for
  loss $\leftarrow \frac{\text{loss}}{B}$ ▷ Compute average loss.
  $\theta \leftarrow \theta - \eta \nabla_\theta$loss ▷ Update network parameters via backpropagation.
end while

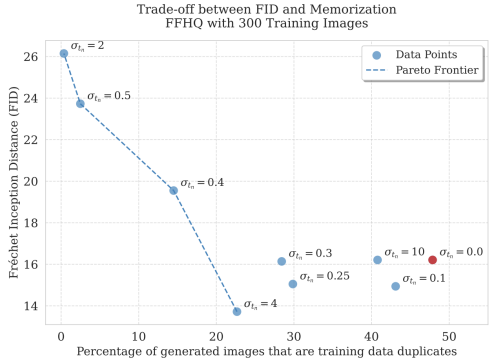


Figure 1: (FID, Memorization) pairs for different values of σ_{t_n} used in our proposed Algorithm 1 (presented in Section 3) for training diffusion models from limited data. The standard DDPM objective corresponds to $\sigma_{t_\text{n}} = 0$ and it is not in the Pareto frontier. Setting σ_{t_n} too low or too high reverts back to the DDPM behavior. Values for $\sigma_{t_\text{n}} \in [0.4, 4]$ strike different balances between memorization and quality of generated images. The models in this Figure are trained on only 300 images from FFHQ.

Figure 3: Taken from Shah et al. (2025). Various instantiations of the algorithm of Shah et al. (2025).

> the algorithm learns the same score function for times $t \leq t_\text{n}$ as DDPM. This contributes to generating samples with high-quality details, copied from the training set.

Table 1: FID and Memorization results comparing DDPM and Algorithm 1. Memorization is measured as DINOv2 similarity between generated samples and their nearest training neighbors. We achieve the same or better FID with significantly lower memorization.

		# Train Images					
		300		1k		3k	
		DDPM	Ours	DDPM	Ours	DDPM	Ours
CIFAR-10	FID	25.1	**23.91**	10.46	**10.36**	14.73	**14.26**
	S>0.9	78.96	**44.84**	75.86	**69.08**	53.40	**52.24**
	S>0.925	67.2	**20.22**	57.98	**47.26**	11.92	**11.36**
	S>0.95	56.56	**9.64**	43.44	**26.34**	0.08	**0.06**
FFHQ	FID	16.21	**15.05**	12.26	**11.3**	**6.42**	6.46
	S>0.85	63.38	**49.68**	55.36	**32.08**	21.58	**20.08**
	S>0.875	55.48	**40.01**	43.82	**17.48**	4.98	**4.53**
	S>0.9	47.86	**29.86**	33.92	**7.52**	0.46	**0.42**
ImageNet	FID	——		50.2	**47.19**	40.66	**39.87**
	S>0.9	——		54.72	**26.68**	32.86	**28.40**
	S>0.925	——		41.66	**15.56**	12.32	**9.44**
	S>0.95	——		25.86	**5.54**	6.08	**4.02**

Figure 4: Taken from Shah et al. (2025). Comparison between the above algorithm and standard DDPM.

References

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR.

Belkin, M., Hsu, D. J., and Mitra, P. (2018). Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. *Advances in neural information processing systems*, 31.

Benton, J., Bortoli, V., Doucet, A., and Deligiannidis, G. (2024). Nearly d-linear convergence bounds for diffusion models via stochastic localization.

Biroli, G., Bonnaire, T., De Bortoli, V., and Mézard, M. (2024). Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957.

Brown, G., Bun, M., Feldman, V., Smith, A., and Talwar, K. (2021). When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, pages 123–132.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. (2023). Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., et al. (2021). Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

Daras, G., Dimakis, A. G., and Daskalakis, C. (2024). Consistent diffusion meets tweedie: Training exact ambient diffusion models with noisy data. *arXiv preprint arXiv:2404.10177*.

Daras, G., Shah, K., Dagan, Y., Gollakota, A., Dimakis, A., and Klivans, A. (2023). Ambient diffusion: Learning clean distributions from corrupted data. In *Thirty-seventh Conference on Neural Information Processing Systems*.

De Bortoli, V. (2022). Convergence of denoising diffusion models under the manifold hypothesis. *arXiv preprint arXiv:2208.05314*.

Dieleman, S. (2024). Diffusion is spectral autoregression.

Feldman, V. (2020). Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Kalai, A. T. and Vempala, S. S. (2024). Calibrated language models must hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 160–171, New York, NY, USA. Association for Computing Machinery.

Kamb, M. and Ganguli, S. (2024). An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*.

Li, M. and Chen, S. (2024). Critical windows: non-asymptotic theory for feature emergence in diffusion models. *arXiv preprint arXiv:2403.01633*.

Ma, S., Bassily, R., and Belkin, M. (2018). The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. In *International Conference on Machine Learning*, pages 3325–3334. PMLR.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. (2023). Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Ross, B. L., Kamkari, H., Wu, T., Hosseinzadeh, R., Liu, Z., Stein, G., Cresswell, J. C., and Loaiza-Ganem, G. (2024). A geometric framework for understanding memorization in generative models. *arXiv preprint arXiv:2411.00113*.

Scarvelis, C., Borde, H. S. d. O., and Solomon, J. (2023). Closed-form diffusion models. *arXiv preprint arXiv:2310.12395*.

Shah, K., Kalavasis, A., Klivans, A. R., and Daras, G. (2025). Does generation require memorization? creative diffusion models using ambient diffusion. *arXiv preprint arXiv:2502.21278*.

Slobin, D. I. (2013). Crosslinguistic evidence for the language-making capacity. In *The crosslinguistic study of language acquisition*, pages 1157–1256. Psychology Press.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. (2022). Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. (2023). Understanding and mitigating copying in diffusion models. *arXiv preprint arXiv:2305.20086*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Zhang, C., Bengio, S., Hardt, M., Mozer, M. C., and Singer, Y. (2019). Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhu, X., Anguelov, D., and Ramanan, D. (2014). Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922.