The Interplay between Memorization and Generalization in Deep Learning

Gintare Karolina Dziugaite

Introduction: The Open Secret of Deep Learning

It is an open secret in modern machine learning: neural networks memorize their training data.[1] We are no longer just theorizing about this; there is striking empirical evidence showing that researchers can reconstruct training images or extract verbatim text snippets directly from trained models. Studies like Haim et al. (2023) have reconstructed recognizable images from standard vision models, while Carlini et al. (2021, 2023) have successfully quantified this phenomenon across neural language models, revealing just how frequently training data can be extracted simply by prompting the model appropriately.

Historically, excessive memorization and generalization were not seen as mutually exclusive. A classical example from the literature is the 1-nearest neighbor classifier, which perfectly memorizes its entire training set yet still converges to bounded generalization error (specifically, twice the Bayes risk). In the context of deep learning, the influential work of Zhang et al. (2017) demonstrated that neural networks trained by Stochastic Gradient Descent (SGD) can easily memorize completely randomized labels. Yet, when trained on true labels, generalization remains good even under nontrivial memorization.

This paradox has been further illuminated by theoretical work on "benign overfitting". Research by Bartlett et al. (2020) on overparameterized linear regression, as well as subsequent work on shallow networks (Frei et al., 2022; Kuo et al., 2023), showed that models can interpolate the training data—memorizing the noise—while still generalizing optimally. In our own past work (Negrea, Dziugaite and Roy 2020), we observed that while uniform convergence might fail for the predictor itself in these regimes, it holds for a surrogate predictor. These findings confirm that memorization can have a benign effect. But this leaves us with a critical question: is memorization *only* a benign side-effect of our overparameterized architectures, or is it a fundamental necessity for learning?

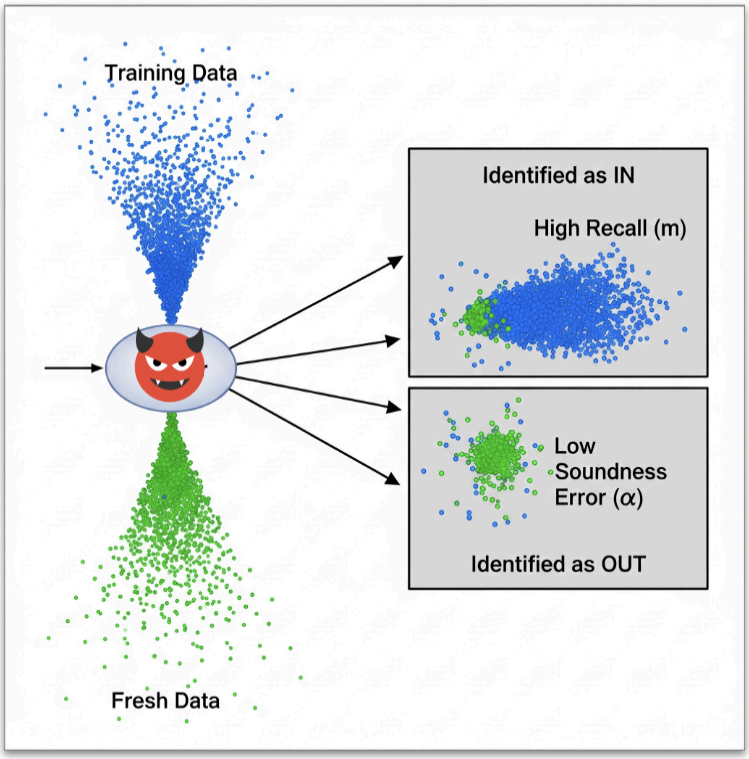
Part 1: Does Learning Require Memorization?

[1] *It's not that a model includes a literal copy of any of its training data, but rather whether a model can be induced to generate near-copies of some training examples when prompted by appropriate techniques and instructions.

The short answer is yes. Theoretical work by Vitaly Feldman (2020) demonstrated that when dealing with long-tailed data distributions—which are ubiquitous in natural, real-world datasets—learning algorithms are actually required to memorize rare examples on average to achieve close-to-optimal generalization.

While Feldman's work establishes this necessity based on the structure of the data distribution, we wanted to investigate if memorization is inextricably linked to the process of learning itself, even in fundamentally simpler environments. In our recent work (Attias, Dziugaite, Haghifam, Livni, and Roy, ICML 2024), we explored the link between optimal learning and memorization within the framework of Stochastic Convex Optimization (SCO). We study SCO because it is mathematically well-understood—we know the optimal learning rates and algorithms—yet it is rich enough to exhibit deep learning phenomena like benign overfitting.

In this work, we formalize memorization through the lens of traceability—a property of the learning algorithm itself, rather than just the training data or a specific attacker. You might be familiar with Membership Inference Attacks (MIAs), where an adversary empirically tries to guess whether a specific data point was used in a model's training set. While MIAs are typically used as practical tools to audit trained models, traceability serves as an information-theoretic counterpart to study this leakage at the algorithm level. It asks: is there *some* learning problem — a specific data distribution — where the algorithm inherently leaks enough information into its final model that a hypothetical adversary could trace back the original training data?



To quantify this, we define a theoretical adversary based on two metrics: it must be α-sound (meaning it has a very low false positive rate, rarely accusing a fresh data point of being from the training set), and it must certify a high recall (successfully identifying at least m actual training examples). If there exists *any* data distribution and adversary that can achieve this for a given learning algorithm, we say that the algorithm is (α, m)-traceable. In short, traceability measures whether an algorithm is

fundamentally susceptible to having its training data reverse-engineered on at least some problems.

Our core finding establishes that **memorization is necessary in high-dimensional SCO**. Specifically, we proved that for any learning algorithm attempting to achieve a small excess error ϵ using n samples in a high-dimensional space ($d > \Omega(n^2 \log n / \alpha)$), there exists a data distribution that forces the algorithm to be traceable. Under these conditions, the adversary is guaranteed to successfully recall $\Omega(1/\epsilon^2)$ training samples.

This reveals a sharp phase transition and a harsh reality regarding the privacy-accuracy tradeoff. By using Conditional Mutual Information (CMI) to measure the dependence between the training data and the learned model, we show that achieving optimal accuracy mathematically guarantees that the model leaks information about its training data. If a practitioner is willing to tolerate higher error, they can employ techniques like Differential Privacy (DP) to act as a shield, building a provably non-traceable model. However, in the low-error regime, where we seek optimal accuracy, traceability is unavoidable. There is no free lunch: optimal accuracy comes at the cost of being traceable.

(Note: While these strict mathematical bounds are proven in the convex setting, empirical evidence suggests these dynamics heavily influence the non-convex optimization of deep neural networks as well.)

Part 2: The Data Diet and Data Influence

If we accept that some memorization is a mathematical necessity for learning the long tail of a distribution, we must also recognize that not all memorization benefits generalization. Relying solely on the premise that more data is better can sometimes degrade a model's performance. Our work on the "Data Diet" (Paul et al., 2021) demonstrates that underfitting—or entirely removing—certain examples can actually improve overall generalization.

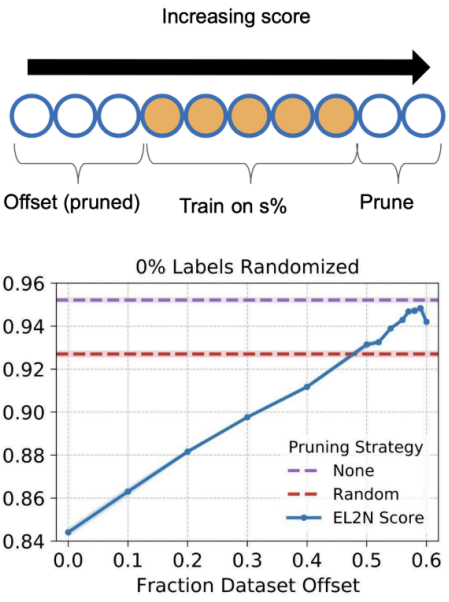
To systematically craft this data diet, we need a way to measure an individual example's influence on the model's final generalization. Historically, finding important training examples required training a model to convergence and utilizing computationally expensive leave-one-out techniques or influence functions. However, we found that it is possible to identify these examples early in the training process.

A practical and computationally efficient metric for this is the EL2N (Error L2-Norm) score. The EL2N score is the L2 distance between the network's predicted probability distribution and the true, one-hot label. We only need to compute this score after a few epochs of training

(e.g., at epoch 10 for a standard ResNet on CIFAR-10). The score measures how much the network struggles to classify a specific example early in the learning trajectory.

When a dataset is sorted by EL2N scores, a spectrum of "difficulty" emerges. Examples with low EL2N scores are the "easy" or typical examples—such as a centered, well-lit picture of a car. The network learns these quickly, and they tend to be redundant; you can prune a large fraction of these low-scoring examples and still match the baseline accuracy of training on the full dataset.

Conversely, examples with high EL2N scores are the "difficult" ones. Visually, these often turn out to be outliers: oddly cropped images, unusual angles, blurry subjects, or incorrectly labeled data. These high-scoring examples have a significant influence on the model's decision boundaries. But high influence does not necessarily mean positive influence.



Because high-scoring examples are difficult, the network is often forced to rote-memorize them to drive the training loss to zero. To test the impact of this, we conducted an experiment where we randomly selected 10% of the training examples and assigned them incorrect labels. When we calculated the early-training EL2N scores, these noisy, corrupted examples clustered at the highest end of the scoring distribution. Since the network eventually achieves perfect training accuracy on these corrupted labels, it indicates they are being memorized.

This led to our next experiment: instead of just keeping the hardest examples, we introduced an "offset" pruning strategy. We pruned the "easy" examples, but we also pruned a small subset of the highest-scoring examples—removing the tail of outliers and mislabeled data.

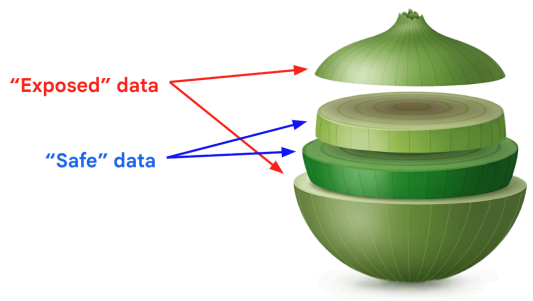
By excluding that subset of the highest-scoring examples, we observed a boost in generalization performance compared to training on the full dataset or a random subset.

The takeaway connects to parallel work we have done exploring pruning through the lens of regularization (Jin et al., 2022). By preventing the network from seeing—and therefore memorizing—the most difficult examples in a dataset, the pruning acts as a targeted, data-dependent regularizer. It allows the network to allocate its capacity toward learning generalizable features rather than memorizing unhelpful noise. Underfitting some examples can sometimes be a mechanism for better generalization.

Part 3: Controlling Memorization: Pre-emptive Layering vs. Post-hoc Surgery

Because we cannot realistically ask for *no* memorization without severely degrading performance or requiring exponentially larger datasets for Differential Privacy , the pragmatic approach is to ask for *controlled* memorization. This brings us to a fascinating design crossroads: do we adapt our training distributions to explicitly dictate what the model learns and memorizes, or do we rely on post-factum modifications to clean up the model after the fact?

The "Reverse Onion" Effect: Pre-emptive Control When we selectively remove data from a trained model, we may observe the "privacy onion effect" (Carlini et al., 2022) —peeling back one layer of memorized data can inadvertently expose other examples, making them *more* memorized or vulnerable. But what if we applied this concept in reverse during training?



Can we intentionally layer our training distributions to achieve a perfectly controlled memorization-generalization profile? By carefully curating the data diet and controlling example frequency, we might be able to construct a curriculum that forces the model to memorize beneficial structural knowledge while discarding spurious, unhelpful outliers. The theoretical appeal here is immense: if we can precisely engineer the input distribution, we can dictate the exact "layers" of the model's memory. However, at the scale of modern foundation models, perfect data curation is incredibly difficult, making pure pre-emptive control an elusive holy grail.

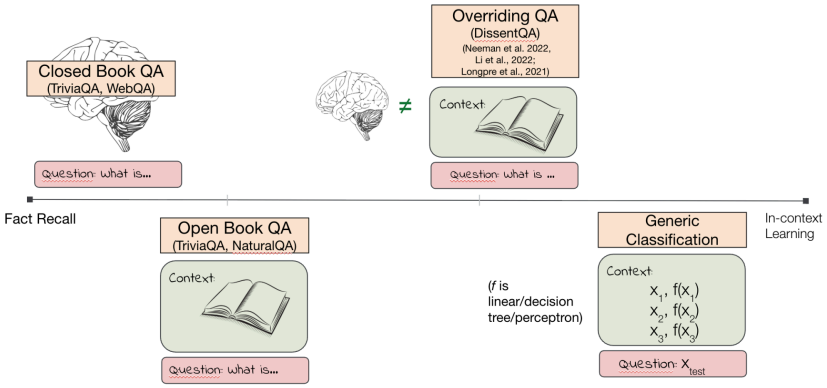
Post-factum Modifications: Unlearning and Pruning When we cannot perfectly layer the onion during training, we must rely on post-hoc tools like unlearning, model editing, and pruning to mitigate unwanted memorization. But as our research shows, these surgical interventions come with their own complex dynamics.

When we apply unlearning, the process is not equally effective for all examples. Our empirical work (Baluta et al., 2024), shows that unlearning memorized outliers—examples that are out-of-distribution—is relatively "easier" and does not damage the model's performance on core tasks. Conversely, unlearning in-distribution samples may be detrimental to performance, and examples with high repetitions in the training set are much more difficult to unlearn.

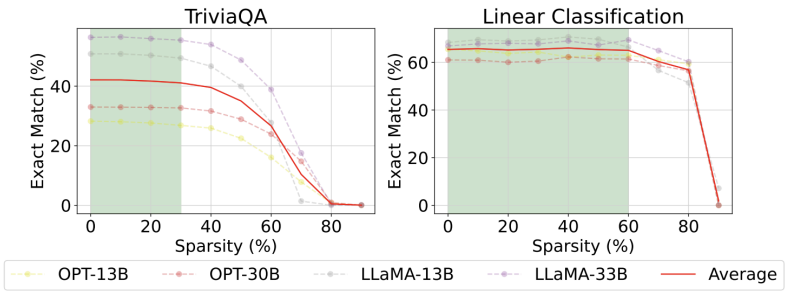
We provided a theoretical backing for these empirical findings by analyzing per-instance privacy loss (Sepahvand et al., 2025). Rather than relying on a single, worst-case DP

guarantee, per-instance privacy loss bounds the Rényi divergence between a model trained with and without a specific data point. We found that this privacy loss perfectly predicts unlearning difficulty. Examples that incurred a high privacy loss during training require significantly "more" unlearning—for instance, requiring more steps of Stochastic Gradient Langevin Dynamics (SGLD)—to successfully erase from the model's weights.

Furthermore, architectural interventions like pruning offer a different kind of post-hoc control. When we scale down Large Language Models (LLMs) via pruning (Jin et al., 2024), we observe a distinct decoupling of capabilities. To measure this, we systematically separated tasks that require retrieving specific training data (such as closed-book TriviaQA) from tasks that require on-the-fly reasoning or learning from a provided prompt (such as open-book QA, overriding QA, or in-context linear classification).



The results showed a clear divergence in how these capabilities degrade. Fact recall—the direct regurgitation of memorized pre-training data—deteriorates quickly, dropping significantly at around 30% sparsity. This suggests that specific factual knowledge is stored in a way that is highly vulnerable to parameter removal. However, In-Context Learning (ICL) and generic reasoning capabilities are remarkably robust, withstanding up to 60-70% pruning before collapsing. This points to ICL either being implemented redundantly across the network or concentrated in regions of the architecture less susceptible to standard pruning.



These findings have practical implications for how we deploy LLMs and manage their memorized data. If pruning disproportionately harms factual memorization rather than reasoning, it can be used intentionally as a mitigation tool. We could heavily prune a model to scrub its internal rote facts—reducing unwanted memorization and improving inference-time compute efficiency—while preserving its core reasoning engine. This "fact-free" reasoning

model could then be paired with an external memory source, such as a Retrieval-Augmented Generation (RAG) system, to retrieve evidence into the context window at inference time, giving us precise control over the information the model uses.

Which Approach Wins? Ultimately, neither approach is a silver bullet on its own. While unlearning and pruning are promising tools to scrub rote facts from a model while preserving its underlying reasoning engine, they are reactive measures. The most robust path forward likely lies in a hybrid approach: using intelligent data diets to lay down a healthy baseline of generalization, while reserving targeted post-hoc modifications for the inevitable instances of unwanted, rigid memorization.

Looking Forward: The Frontier of Memorization Control

The dynamics of memorization sit at the core of how deep learning works. While overparameterization might somewhat mitigate the negative impacts of memorizing noise, relying purely on scale is costly and inefficient. Furthermore, the scaling law approach to data mixture optimization fundamentally fails to model the nuanced, compounding interactions between datasets. Memorization is both a bug and a feature; the next era of deep learning will be defined by how precisely we can control it.

This brings us to one of the most pressing open questions in the field: **Should we focus our efforts on pre-emptive data engineering, or on post-training model surgery?** Can we practically build the "reverse privacy onion"—curating data mixtures and training curriculums so perfectly that we dictate exactly what structural knowledge the model memorizes from day one? Or is the entropy of internet-scale data simply too vast, meaning we will *always* have to rely on reactive interventions like unlearning, model editing, and targeted pruning to clean up unwanted facts and rigid biases after training is complete?

The answer likely lies in a hybrid approach, but finding that exact balance requires solving several theoretical and empirical puzzles. Theoretically, we still need to precisely map the mathematical tradeoff between learning and memorization , and understand what the strict necessity to memorize means for the theoretical limits of machine unlearning. Empirically, we need better tools to predict *ahead of time* which examples in a massive dataset are most likely to be memorized. Finally, exploring architectural changes—such as modularity or retrieval-augmented generation (RAG)—that can separate rigid information storage from flexible reasoning remains a critical frontier.