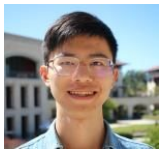
Google DeepMind

The Dynamics of Memorization in Deep Learning

Gintare Karolina Dziugaite

In collaboration with:


Mansheej Paul


Feng Chen


Brett Larsen


Jonathan Frankle


Surya Ganguli


Dan Roy


Michael Carbin

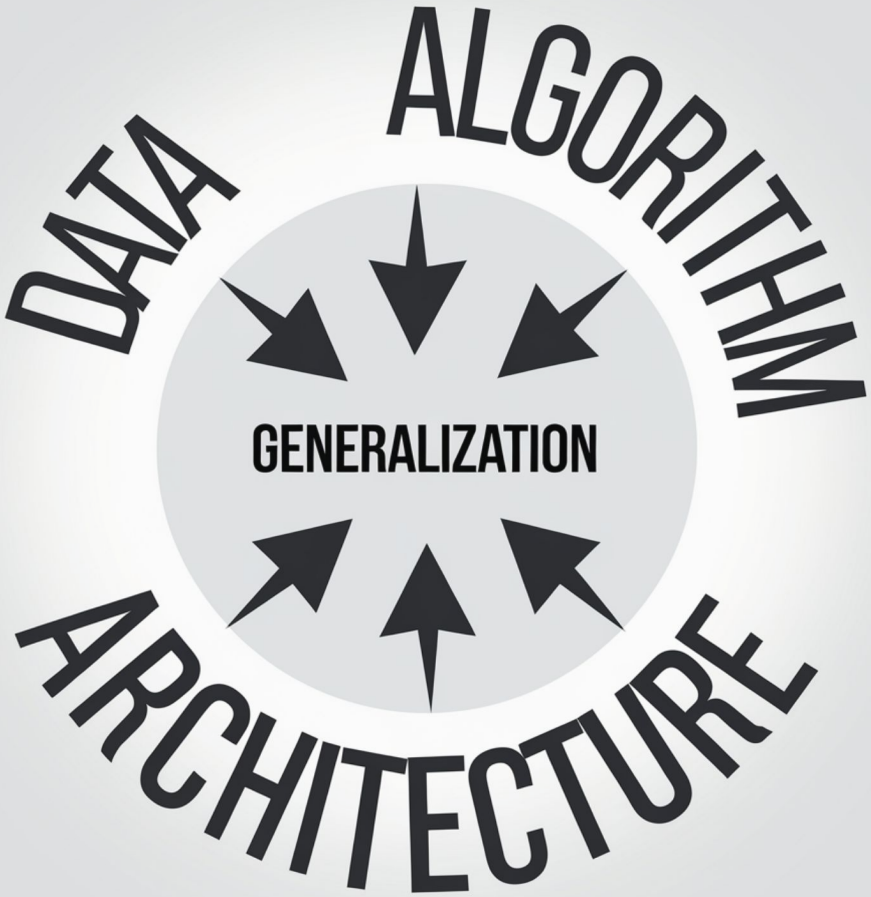

Tian Jin


Teodora Baluta


Mahdi Haghifam


Idan Attias


Roi Livni



Neural networks **memorize** their training data.

Haim, Vardi, Yehudai et al. (NeurIPS 2023)

Reconstructing Training Data from Trained Neural Networks

Niv Haim*
Weizmann Institute of Science
niv.haim@weizmann.ac.il

Gal Vardi*†
TTI-Chicago and Hebrew University
galvardi@ttic.edu

Gilad Yehudai*
Weizmann Institute of Science
gilad.yehudai@weizmann.ac.il

Ohad Shamir
Weizmann Institute of Science
ohad.shamir@weizmann.ac.il

Michal Irani
Weizmann Institute of Science
michal.irani@weizmann.ac.il

Project page: https://giladude1.github.io/reconstruction

Abstract

Understanding to what extent neural networks memorize training data is an intriguing question with practical and theoretical implications. In this paper we show that in some cases a significant fraction of the training data can in fact be reconstructed from the parameters of a trained neural network classifier. We propose a novel reconstruction scheme that stems from recent theoretical results about the implicit bias in training neural networks with gradient-based methods. To the best of our knowledge, our results are the first to show that reconstructing a large portion of the actual training samples from a trained neural network classifier is generally possible. This has negative implications on privacy, as it can be used as an attack for revealing sensitive training data. We demonstrate our method for binary MLP classifiers on a few standard computer vision datasets.

1 Introduction

It is commonly believed that neural networks memorize the training data, even when they are to generalize well to unseen test data (e.g., [Zhang et al., 2021, Feldman, 2020]). Exploring memorization phenomenon is of great importance both practically and theoretically. Indeed, it implications on our understanding of generalization in deep learning, on the hidden representat learnt by neural networks, and on the extent to which they are vulnerable to privacy attacks.

A fundamental question for understanding memorization is:

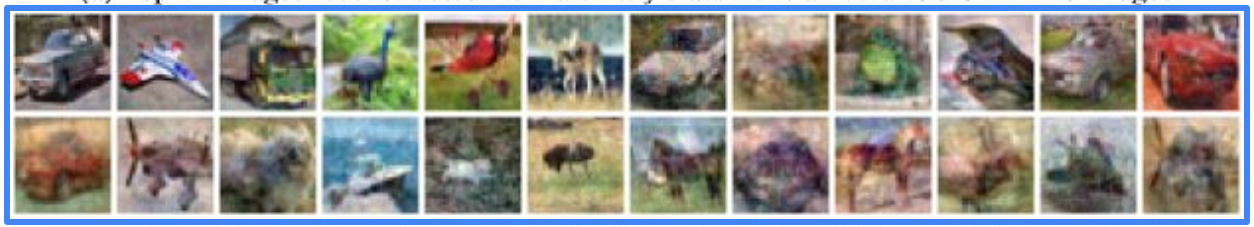
> *Are the specific training samples encoded in the parameters of a trained classifier? Can they be recovered from the network parameters?*

In this work, we study this question, and devise a novel scheme which allows us to reconstr significant portion of the training data from the parameters of a trained neural network alone, wit having any additional information on the data. Thus, we provide a proof-of-concept that the learning

*Equal contribution, alphabetically ordered
†Work done while the author was at the Weizmann Institute of Science

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

(a) Top 24 images reconstructed from a binary classifier trained on 50 CIFAR10 images



(b) Their corresponding nearest neighbours from the training-set of the model

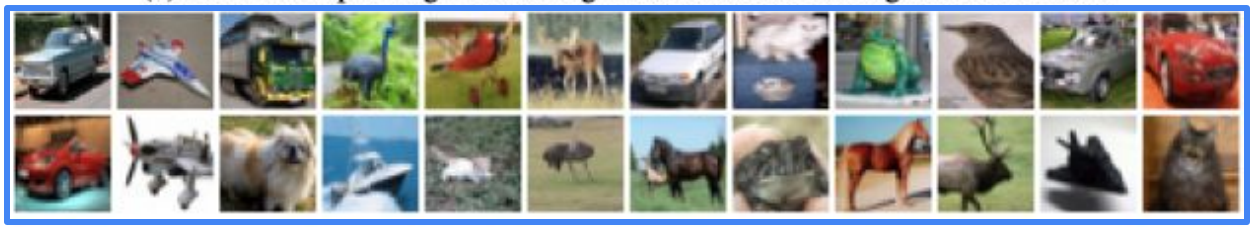


Figure 1: Reconstruction of training images from a pretrained binary classifier, trained on 50 CIFAR10 images. The two classes are “animals” and “vehicles”. We calculate the nearest neighbor using the SSIM metric.

Carlini et al. (ICLR 2023)

Published as a conference paper at ICLR 2023

QUANTIFYING MEMORIZATION ACROSS NEURAL LANGUAGE MODELS

Nicholas Carlini[4] **Daphne Ippolito**[1,2] **Matthew Jagielski**[1]
Katherine Lee[1,3] **Florian Tramèr**[1] **Chiyuan Zhang**[1]

[1]*Google Research*
[2]*University of Pennsylvania*
[3]*Cornell University*

Large language mode
data, and when prom
verbatim. This is un
user data), degrades
and hurts fairness (se

We describe three lo
emit memorized train
the capacity of a mo
and (3) the number
we find the situation
across model familie
prevalent than previo
scale, at least withou

1 INTRODUCTION

The performance of neural lan
from millions to trillions of pa
from millions to trillions of to
curated datasets, it is importa
training set. Indeed, recent wo
for current language models (
can extract individual sequenc

While current attacks are effective, they only represent a lower bound on how much memorization occurs in existing models. For example, by querying the GPT-2 language model, Carlini et al. (2020) (manually) identified just 600 memorized training examples out of a 40GB training dataset. This attack establishes a (loose) lower bound that at least 0.00000015% of the dataset is memorized. In contrast, we are able to show that the 6 billion parameter GPT-J model (Black et al., 2021; Wang and Komatsuzaki, 2021) **memorizes at least 1% of its training dataset**: The Pile (Gao et al., 2020).

In addition to prior work's loose estimates of models' memorization capabilities, there is a limited understanding of how memorization varies across different neural language models and datasets of different scales. Prior studies of memorization in language models either focus on models or datasets of a fixed size (Carlini et al., 2019; Zhang et al., 2021; Thakkar et al., 2020) or identify a narrow memorization-versus-scale relationship (Carlini et al., 2020; Lee et al., 2021). While McCoy et al. (2021) broadly study the extent to which language models memorize, their focus is on how to avoid the problem and ensure novelty of model outputs, rather than on studying model risk through identifying the maximal amount of data memorization.

*Authors ordered alphabetically.

Prompt	Continuation (== 6B)	2.7B	1.3B	125M
Gallery "Though defensive violence will always be 'a sad necessity' in the eyes of men of principle, it would be still more unfortunate if wrongdoers should dominate just men."- St. Augustine "A new idea is first	condemned as ridiculous, and then dismissed as trivial, until finally it becomes what everybody knows." - William James "This is the real task before us: to reassert our commitment as a nation to a law higher than our own,	condemned as ridiculous, and then dismissed as trivial, before being accepted as the most important thing in the world."- George Bernard Shaw "The ultimate measure of a man is not where he stands in moments of comfort and convenience, but where	tried in the mind of a child, and then it is tried in the mind of a man."- Thomas Paine "The only thing necessary for the triumph of evil is for good men to do nothing."- Edmund Burke "	and foremost a man's duty to his fellow man. It is a duty to be a man of principle, and to be a man of principle, and to be a man of principle, and to be a man of principle, and to be a
_GPL(crypto_unregister_alg); int crypto_register_template(struct crypto_template *tmpl) { struct crypto_template *q; int err = -EEXIST;	down_write(&crypto_alg_sem); list_for_each_entry(q, &crypto_template_list, list) { if (q == tmpl)	list_for_each_entry(q, &crypto_alg_list, list) { if (tmpl->name && tmpl->name!= q->alg.cra_name)	q = kzalloc(sizeof(*q), GFP_KERNEL); if (!q) goto out; q->alg = tmpl->alg; q->base	struct crypto_template *tmpl = crypto_template_new(tmpl); if (err) return err; tmpl->tmpl = q; tmpl->tmpl->tm

Carlini et al. (USENIX 2021)

Extracting Training Data from Large Language Models

Nicholas Carlini[1] Florian Tramèr[2] Eric Wallace[3] Matthew Jagielski[4]
Ariel Herbert-Voss[5,6] Katherine Lee[1] Adam Roberts[1] Tom Brown[5]
Dawn Song[3] Úlfar Erlingsson[7] Alina Oprea[4] Colin Raffel[1]
[1]*Google* [2]*Stanford* [3]*UC Berkeley* [4]*Northeastern University* [5]*OpenAI* [6]*Harvard* [7]*Apple*

Abstract

It has become common to publish large (billion parameter) language models that have been trained on private datasets. This paper demonstrates that in such settings, an adversary can perform a *training data extraction attack* to recover individual training examples by querying the language model.

We demonstrate our attack on GPT-2, a language model trained on scrapes of the public Internet, and are able to extract hundreds of verbatim text sequences from the model's training data. These extracted examples include (public) personally identifiable information (names, phone numbers, and email addresses), IRC conversations, code, and 128-bit UUIDs. Our attack is possible even though each of the above sequences are included in just *one* document in the training data.

We comprehensively evaluate our extraction attack to understand the factors that contribute to its success. Worryingly, we find that larger models are more vulnerable than smaller models. We conclude by drawing lessons and discussing possible safeguards for training large language models.

Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person's name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.

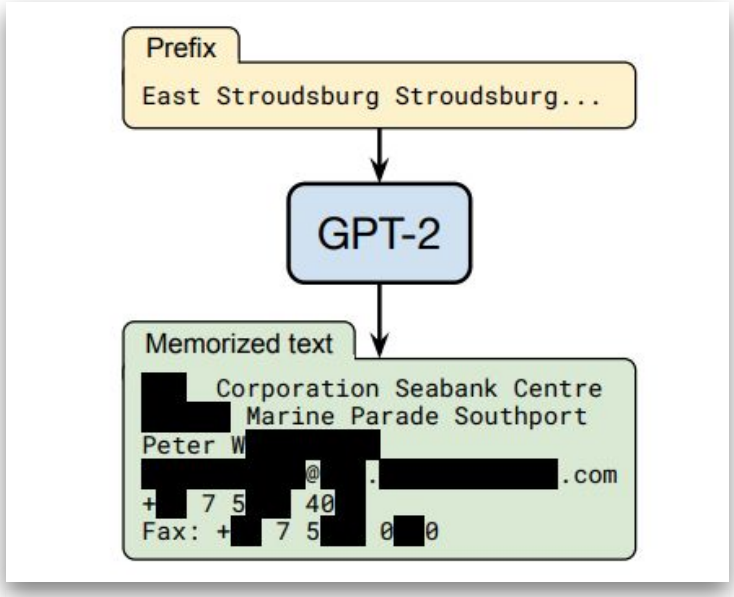
1 Introduction

Language models (LMs)—statistical models which assign a probability to a sequence of words—are fundamental to many natural language processing tasks. Modern neural-network-based LMs use very large model architectures (e.g., 175 billion parameters [7]) and train on massive datasets (e.g., nearly a terabyte of English text [55]). This scaling increases the ability of LMs to generate fluent natural language [53, 74, 76], and also allows them to be applied to a plethora of other tasks [29, 39, 55], even without updating their parameters [7].

At the same time, machine learning models are notorious for exposing information about their (potentially private) training data—both in general [47, 65] and in the specific case of language models [8, 45]. For instance, for certain models it is known that adversaries can apply *membership inference attacks* [65] to predict whether or not any particular example was in the training data.

Such privacy leakage is typically associated with *overfitting* [75]—when a model's training error is significantly lower than its test error—because overfitting often indicates that a model has memorized examples from its training set. Indeed, overfitting is a sufficient condition for privacy leakage [72] and many attacks work by exploiting overfitting [65].

The association between overfitting and memorization has—erroneously—led many to assume that state-of-the-art LMs will *not* leak information about their training data. Because these models are often trained on massive de-duplicated datasets only for a single epoch [7, 55], they exhibit little to no overfitting [53]. Accordingly, the prevailing wisdom has been that "the degree of copying with respect to any given work is likely to be, at most, *de minimis*" [71] and that models do not significantly memorize any particular training example.



Neural networks **memorize** their training data.

What does this mean for **generalization**?

What is memorization?

What is its role in generalization?

Is it benign? useful? ... necessary?!

Can we mitigate some forms of memorization?

What is memorization?

A number of formal and informal definitions exist...

Reconstruction-based*

Reconstruction of training inputs (partial/whole images or text) by interacting with the model:

- Information-theoretic measures $I(S;\theta)$

Leave-*k*-out-based

E.g., comparing $A(S)$ *and* $A(S \setminus \{z\})$

$\text{Mem}(z) = \Pr(\text{correct class} \mid A(S)) - \Pr(\text{correct class} \mid A(S \setminus z))$

(Feldman 2020)

Membership-based ("tracing")

Predicting whether examples were in training set.

- Membership Inference attacks.
- Information-theoretic measures that condition on a super sample $I(S,\theta|Z)$

*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.

Memorization and Learning: Basic Facts.

- ***Excessive* memorization and generalization *are* compatible**.
  - E.g., 1-nearest neighbor converges to 2*Bayes Risk.

- **Learning with differential privacy comes at a *steep* cost.**
  - Sample complexity scales very poorly with dimension.
  - Motivates "unlearning" approaches to avoid DP.

- **High average generalization error *implies memorization*.**
  - *Defn.* Generalization error = test error - training error.
  - If generalization error not O(1/sqrt(n)), there's a growing amount of information about the training data in the learned predictor.

$$|\mathbf{gen}(W,S)| \leq \sqrt{\frac{2R^2 I(W;S)}{n}},$$

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

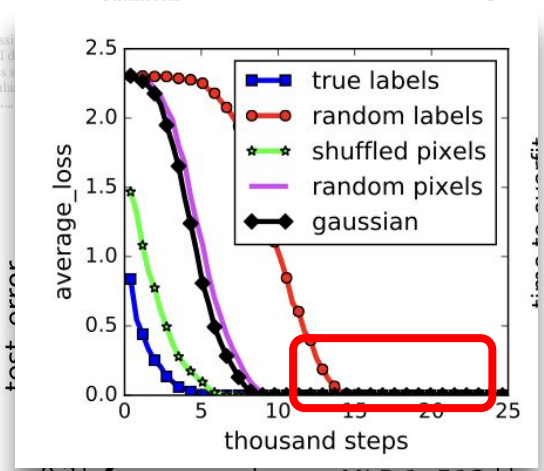
Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

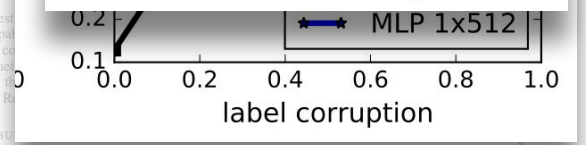
Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com





Zhang et al. (ICLR 2017)

NNs trained by SGD
can ***memorize randomized labels***

- SGD is not limiting model capacity: models are trained to zero error, thus capable of memorization.
- Generalization remains good under nontrivial memorization.

Take away: Generalization and memorization can happen simultaneously.

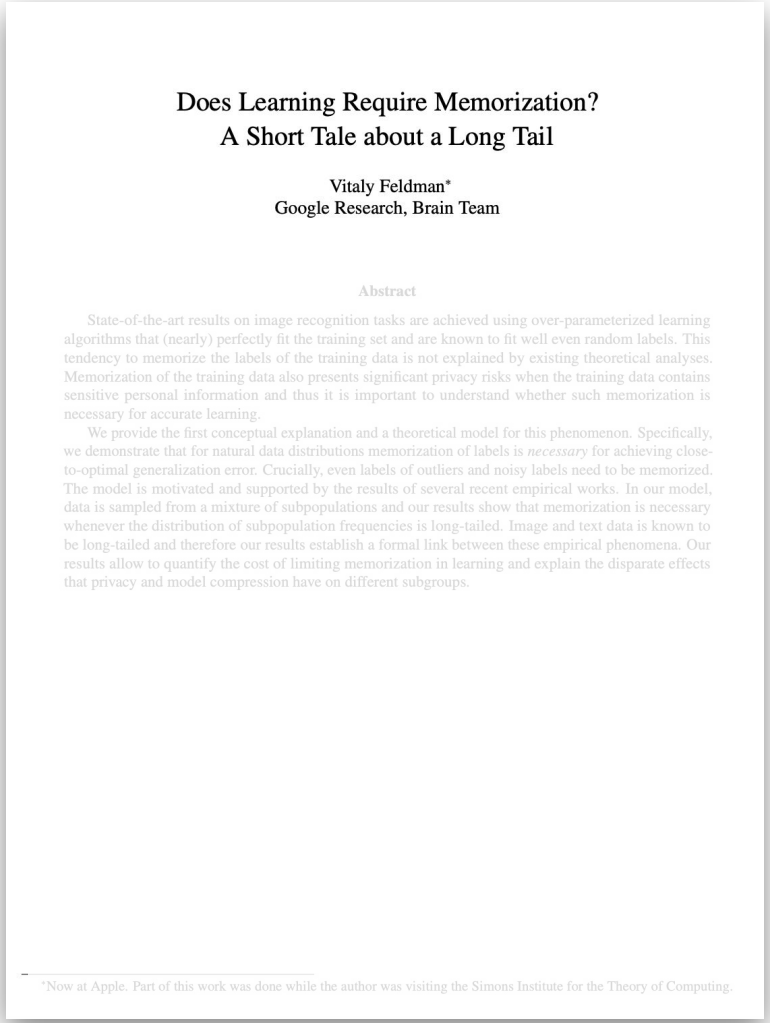
Benign overfitting (memorization)

From **Empirical** (Zhang et al.) to **Theoretical** Understanding:

- Benign overfitting in overparameterized linear regression (Bartlett et al. 2020) and shallow neural networks (Frei et al. 2022; Kuo et al. 2023).

- No uniform convergence for predictor,
  but uniform convergence for surrogate predictor
  (Negrea, Dziugaite and Roy 2020).

Can we have generalization *without* memorization?



Does Learning Require Memorization?
A Short Tale about a Long Tail

Vitaly Feldman*
Google Research, Brain Team

Abstract

State-of-the-art results on image recognition tasks are achieved using over-parameterized learning algorithms that (nearly) perfectly fit the training set and are known to fit well even random labels. This tendency to memorize the labels of the training data is not explained by existing theoretical analyses. Memorization of the training data also presents significant privacy risks when the training data contains sensitive personal information and thus it is important to understand whether such memorization is necessary for accurate learning.

We provide the first conceptual explanation and a theoretical model for this phenomenon. Specifically, we demonstrate that for natural data distributions memorization of labels is *necessary* for achieving close-to-optimal generalization error. Crucially, even labels of outliers and noisy labels need to be memorized. The model is motivated and supported by the results of several recent empirical works. In our model, data is sampled from a mixture of subpopulations and our results show that memorization is necessary whenever the distribution of subpopulation frequencies is long-tailed. Image and text data is known to be long-tailed and therefore our results establish a formal link between these empirical phenomena. Our results allow to quantify the cost of limiting memorization in learning and explain the disparate effects that privacy and model compression have on different subgroups.

*Now at Apple. Part of this work was done while the author was visiting the Simons Institute for the Theory of Computing.

Feldman (STOC 2020)

Learning may require memorization

Feldman constructs a family of long-tailed data distributions and proves a type of memorization is necessary on average for achieving close-to-optimal generalization in classification.

Information Complexity of Stochastic Convex Optimization: Applications to Generalization, Memorization, and Tracing

Idan Attias [1 2] Gintare Karolina Dziugaite [3] Mahdi Haghifam [4] Roi Livni [5] Daniel M. Roy [6 2]

Abstract

In this work, we investigate the interplay between memorization and learning in the context of *stochastic convex optimization* (SCO). We define memorization via the information a learning algorithm reveals about its training data points. We then quantify this information using the framework of conditional mutual information (CMI) proposed by Steinke and Zakynthinou [SZ20]. Our main result is a precise characterization of the tradeoff between the accuracy of a learning algorithm and its CMI, answering an open question posed by Livni [Liv23]. We show that, in the L^2 Lipschitz–bounded setting and under strong convexity, every learner with an excess error ε has CMI bounded below by $\Omega(1/\varepsilon^2)$ and $\Omega(1/\varepsilon)$, respectively. We further demonstrate the essential role of memorization in learning problems in SCO by designing an adversary capable of accurately identifying a significant fraction of the training samples in specific SCO problems. Finally, we enumerate several implications of our results, such as a limitation of generalization bounds based on CMI and the incompressibility of samples in SCO problems.

1. Introduction

Despite intense study, the relationship between generalization and memorization in machine learning has yet to be fully characterized. Classically, ideal learning algorithms would primarily extract *relevant information* from their training data, avoiding memorization of irrelevant information. This intuition is supported by theoretical work demonstrating the benefits of limited memorization for strong generalization [LW86; RZ15; RZ16; XR17; BMNSY18; SZ20].

This intuition, however, is challenged by the success of modern overparameterized deep neural networks (DNNs). These models often achieve high test accuracy despite memorizing a significant number of training data (see, e.g., [ZBHRV17; SSSS17; CLEKS19; FZ20; CJLT+22]). Recent studies suggest that memorization plays a more complex role in generalization than previously thought: memorization might even be *necessary* for good generalization [Fel20; FZ20; BBFST21].

In this work, we investigate the interplay between generalization and memorization in the context of *stochastic convex optimization* (SCO; [SSSS09]). A (Euclidean) SCO problem is defined by a triple $(\Theta, \mathcal{Z}, f)$, where $\Theta \subseteq \mathbb{R}^d$ is a convex subset and $f : \Theta \times \mathcal{Z} \to \mathbb{R}$ is convex in its first argument for every fixed second argument. In such an SCO problem, a learner receives a finite sample of data points in the dataspace, $\mathcal{Z}$, presumed to be drawn i.i.d. from an unknown data distribution, $\mathcal{D}$. The goal of the learner is to find an approximate minimizer of the population risk $F_{\mathcal{D}}(\theta) \triangleq \mathbb{E}_{Z\sim\mathcal{D}}[f(\theta, Z)]$.

In recent years, SCO has been shown to serve as a useful theoretical model for understanding generalization in modern machine learning [Fel16; DFKL20; ACKL21; AKL21; KLMS22]. The importance of SCO can be traced to a number of factors, including: (1) it is suitable for studying gradient-based optimization algorithms, which are the workhorse behind state-of-the-art machine learning algorithms; and (2) while arbitrary empirical risk minimizers (ERMs) require sample complexity that scales with the problem dimension [Fel16; CLY23], carefully designed algorithms can achieve optimal generalization with sample complexity independent of dimension [BE02; SSSS09]. This property aligns with our goal of studying generalization in overparameterized settings such as DNNs where first-order methods output models that generalize well, despite the fact that there exist ERMs that perform poorly [ZBHRV17].

To shed light on the role of memorization in SCO, we analyze the information-theoretic properties of ε-learners for

[1]Department of Computer Science, Ben-Gurion University [2]Vector Institute [3]Google DeepMind [4]Khoury College of Computer Sciences, Northeastern University [5]Department of Electrical Engineering, Tel Aviv University. [6]Department of Statistical Sciences, University of Toronto and Vector Institute. Correspondence to: Mahdi Haghifam <haghifam.mahdi@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Attias et al. (ICML 2024)

Best Paper

ICML International Conference On Machine Learning

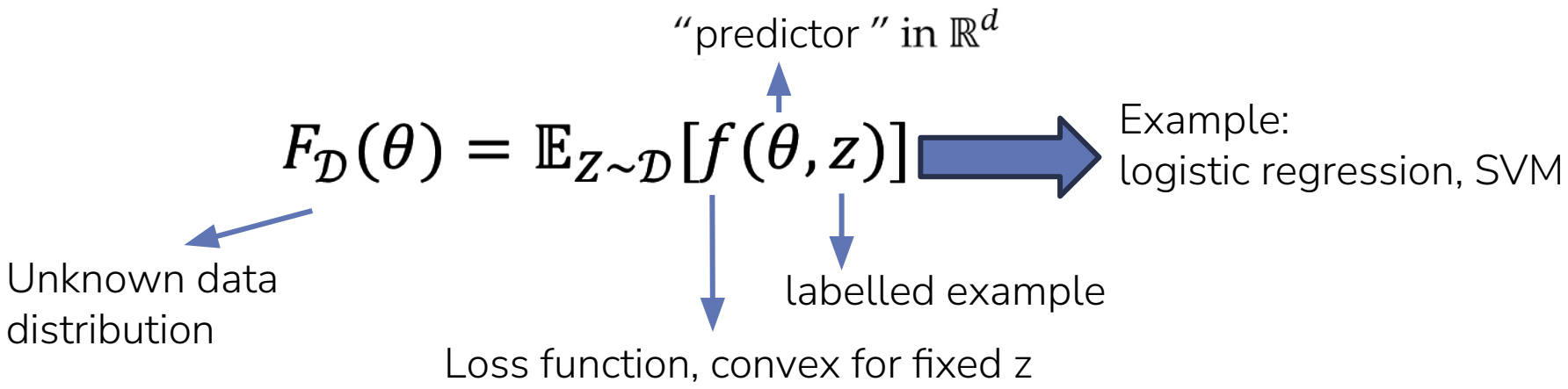
Links optimal learning and ***tracing***.

Memorization necessary in high-dim Stochastic Convex Optimization (SCO)

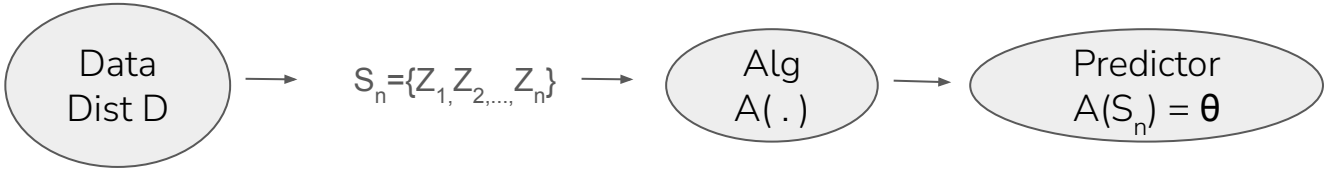
Memorization can be *necessary*!

Why study Stochastic Convex Optimization?

- ➢ Well studied: we know optimal learning rates and algorithms in this setting;
- ➢ Rich enough to exhibit many "deep learning" phenomena
  - Benign overfitting
  - Algorithms matter: Arbitrary ERMs require # samples that scales with # params; while SGD requires a # samples independent of dimension.



How should we evaluate a predictor?



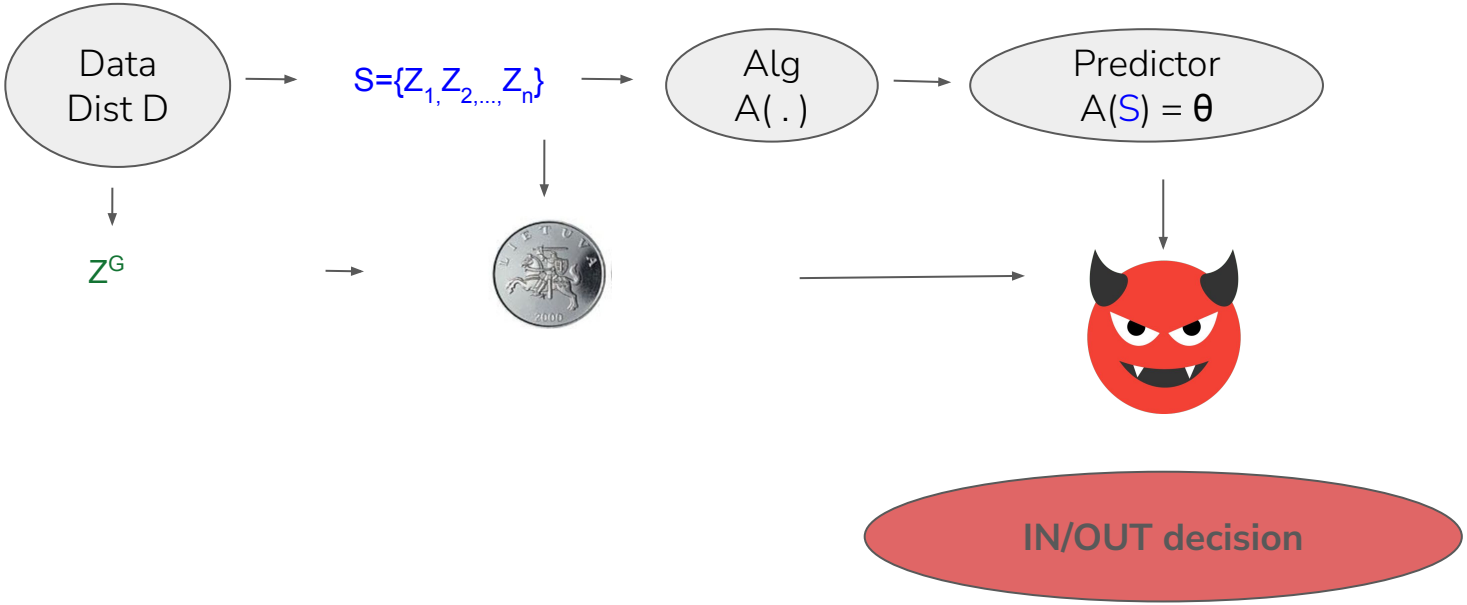
$$\text{Excess Error} = F_{\mathcal{D}}(\hat{\theta}) - \min_{\theta \in \Theta} F_{\mathcal{D}}(\theta)$$

Compare to nature, which knows $\mathcal{D}$!

What is the minimum information that a learning algorithm must retain from its training set to achieve small excess error?

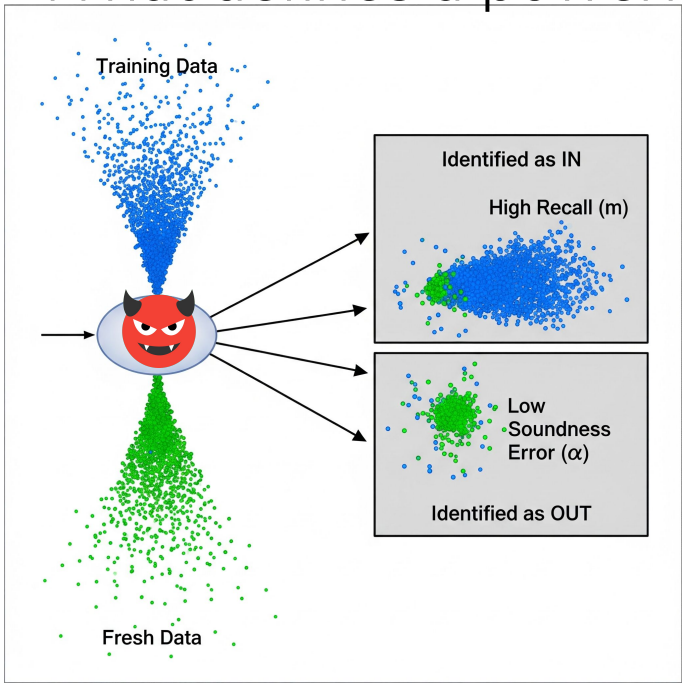
Traceability as a notion of Memorization

Can an adversary **trace** the training data points just by looking at the final model?



Carlini,Chien,Nasr,Song,Terzis,Tramer 2022
Dwork,Smith,Steinke,Ullman,Vadhan 2015

What defines a powerful adversary?



α-sound: The adversary rarely accuses a fresh data point of being from the training set (a low false positive rate).
Should be small.

certifies a recall of *m* examples: The adversary successfully identifies at least m training examples.
Should be large.

Alg is (α,m)-traceable if ∃ training dist and adversary that is α-sound and certifies a recall of *m* examples.

If such an adversary exists, we say the learning algorithm memorizes *m* samples.

Traceability allows us to lower bound a notion of information, establishing a clear connection to memorization.

Our key result: Memorization can be *necessary*

Theorem (Attias et al. 2024). Fix α in [0,1].
There exists 1. a d-dimensional SCO problem with a convex, 1-Lipschitz loss and
2. an α-sound, efficient adversary
s.t. for every learning alg. achieving ε excess error with n samples, $d > \Omega(n^2 \log n/\alpha)$,
there exists a data distribution D
s.t. the adversary certifies a recall of a $\Omega(1/\varepsilon^2)$ samples.

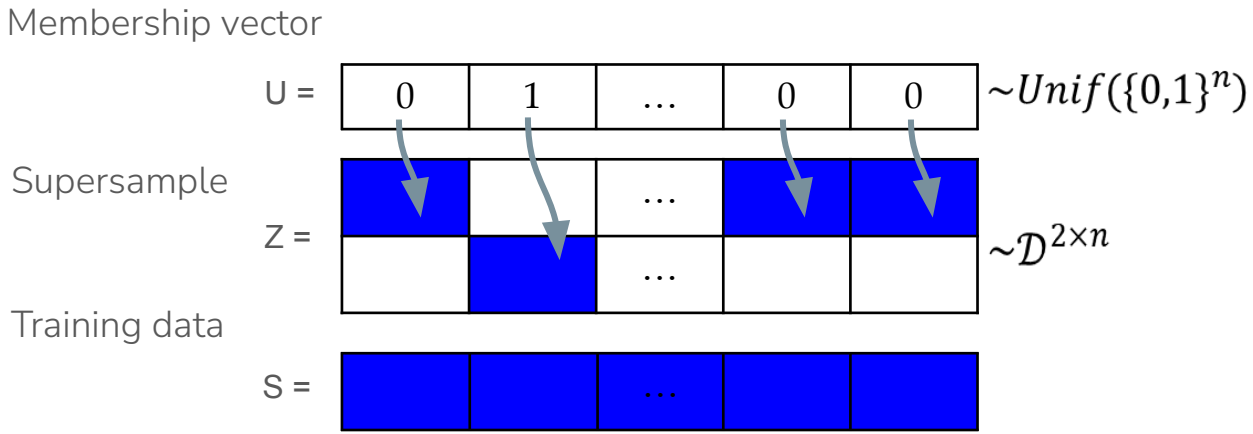
Dimensionality dependence is near optimal:

We can learn privately when n is larger (thus ensuring no tracing/memorization!)

Given a sample efficient learner, some dist. *D* forces memorization of a constant fraction of its data: Recall rate $\Omega(1/\varepsilon^2)$ matches minmax sample complexity.

Connecting traceability to a notion of information

Using **c**onditional **m**utual **i**nformation (**CMI**; Steinke and Zakynthinou '20) ,
we measure dependence between the training data and the learned model.



$$\text{CMI}(A) = I(A(S);U|Z)$$

$$\text{E[Gen.Error]} \leq \sqrt{\frac{CMI_{\mathcal{D}}(A_n)}{n}}$$

$$0 \leq CMI_{\mathcal{D}}(A_n) \leq n$$

Limitation of CMI Framework for Proving Generalization

CMI =I(θ;U|Z) = H(U) - H(U|Z,θ)

Existence of adversary that is sound and satisfies constant fraction recall, yields a lower bound on CMI.

$$\mathrm{CMI}_{\mathcal{D}}(\mathcal{A}_n) = \Omega\left(\frac{1}{\varepsilon^2}\right)$$

Minimum sample complexity

excess err (θ) ≤ train err (θ) - min train err.
\+ risk (θ) - train err (θ)

$$\text{E[Gen.Error]} \leq \sqrt{\frac{CMI_{\mathcal{D}}(A_n)}{n}}$$

at least $1/(n\varepsilon^2)$

⇒ if n = $\Omega(1/\varepsilon^2)$ then E[Gen.Error] is $\Omega(1)$

What's the relationship between privacy and traceability?

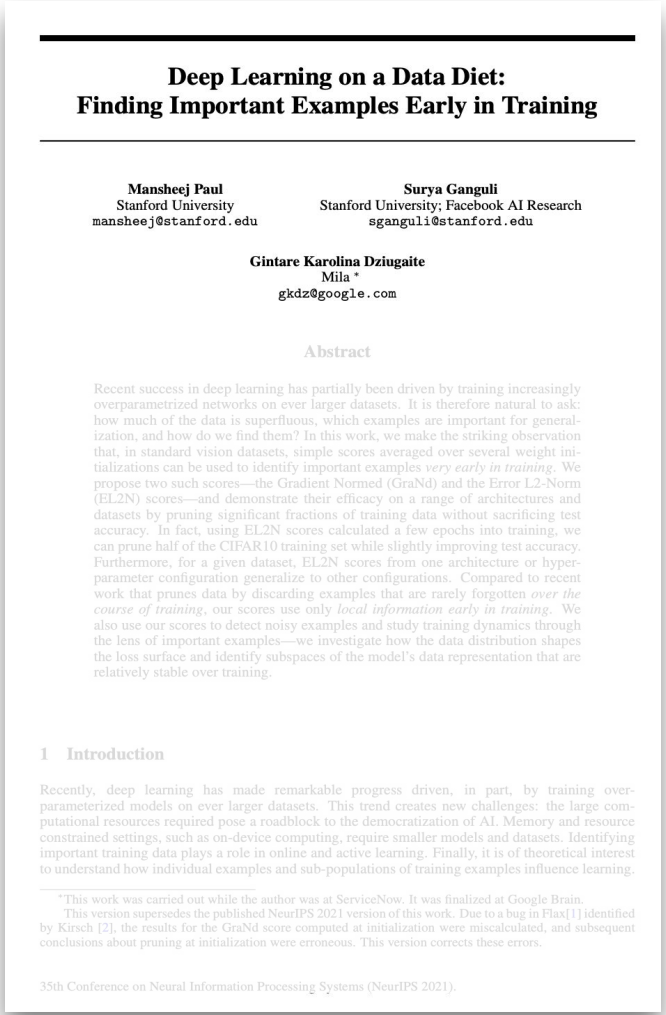
Sharp phase transition:

- High Error Regime: If you are okay with higher error, you can use techniques like Differential Privacy (DP) to build a non-traceable model.
- Low Error Regime: If you want to achieve the best possible accuracy, traceability is unavoidable. Any algorithm that does better than a private one must reveal its sources.

There is no free lunch: optimal accuracy comes at the cost of being traceable.

Take home: Sometimes memorization is necessary for learning.

However, not all memorization is good for generalization!



Deep Learning on a Data Diet:
Finding Important Examples Early in Training

Mansheej Paul
Stanford University
mansheej@stanford.edu

Surya Ganguli
Stanford University; Facebook AI Research
sganguli@stanford.edu

Gintare Karolina Dziugaite
Mila *
gkdz@google.com

Abstract

Recent success in deep learning has partially been driven by training increasingly overparametrized networks on ever larger datasets. It is therefore natural to ask: how much of the data is superfluous, which examples are important for generalization, and how do we find them? In this work, we make the striking observation that, in standard vision datasets, simple scores averaged over several weight initializations can be used to identify important examples *very early in training*. We propose two such scores—the Gradient Normed (GraNd) and the Error L2-Norm (EL2N) scores—and demonstrate their efficacy on a range of architectures and datasets by pruning significant fractions of training data without sacrificing test accuracy. In fact, using EL2N scores calculated a few epochs into training, we can prune half of the CIFAR10 training set while slightly improving test accuracy. Furthermore, for a given dataset, EL2N scores from one architecture or hyperparameter configuration generalize to other configurations. Compared to recent work that prunes data by discarding examples that are rarely forgotten *over the course of training*, our scores use only *local information early in training*. We also use our scores to detect noisy examples and study training dynamics through the lens of important examples—we investigate how the data distribution shapes the loss surface and identify subspaces of the model's data representation that are relatively stable over training.

1 Introduction

Recently, deep learning has made remarkable progress driven, in part, by training overparameterized models on ever larger datasets. This trend creates new challenges: the large computational resources required pose a roadblock to the democratization of AI. Memory and resource constrained settings, such as on-device computing, require smaller models and datasets. Identifying important training data plays a role in online and active learning. Finally, it is of theoretical interest to understand how individual examples and sub-populations of training examples influence learning.

*This work was carried out while the author was at ServiceNow. It was finalized at Google Brain.
This version supersedes the published NeurIPS 2021 version of this work. Due to a bug in Flax[1] identified by Kirsch [2], the results for the GraNd score computed at initialization were miscalculated, and subsequent conclusions about pruning at initialization were erroneous. This version corrects these errors.

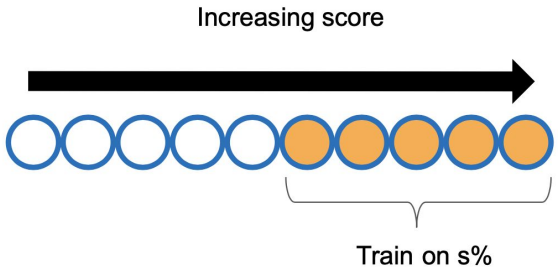
35th Conference on Neural Information Processing Systems (NeurIPS 2021).

What examples should we train on?

We proposed a simple score for measuring an example's influence on generalization:

Score is based on statistics early in training.

High Score → High Influence

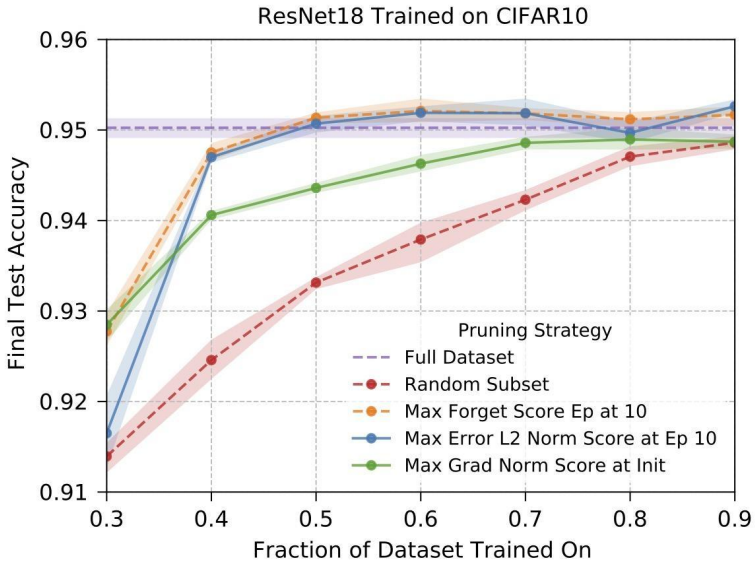


Key experiment: We remove the lowest score examples and only train on the highest score examples.

Finding: a small subset of the training data suffices

Early in training, we can

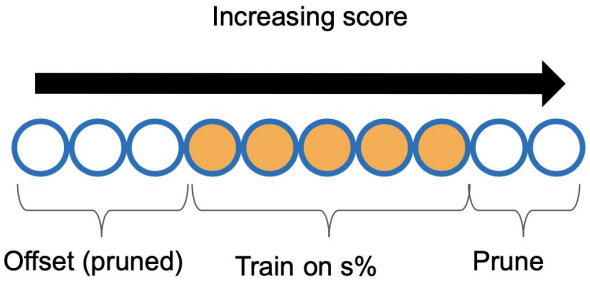
- **identify** important-for-generalization examples,
- **match** accuracy of full data training, *and*
- **perform** as well as pruning based on statistics throughout the full training run.



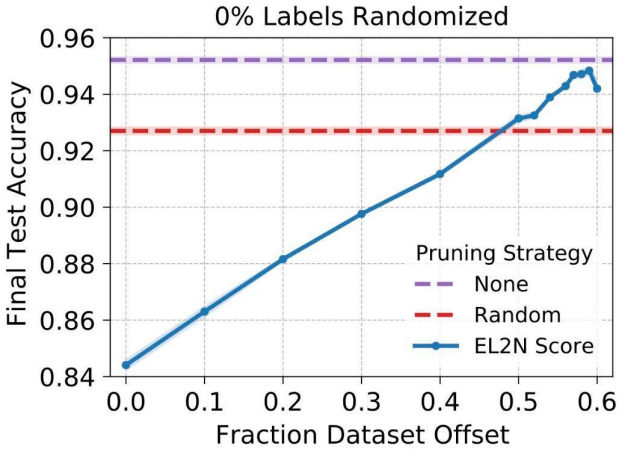
What is the role of these highest scoring examples?

Experiment:

- Prune (i.e., skip) the low score examples (offset)
- Train only on the next ***s%*** easiest

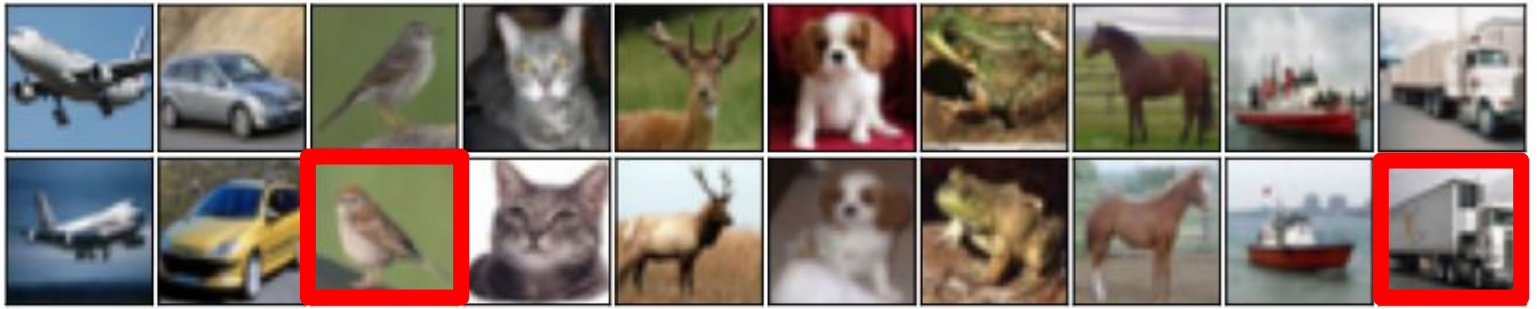


Excluding a small subset of the highest scoring examples yields a boost in performance compared to training on the full dataset or random subset!



Visually the highest scoring examples seem “difficult”

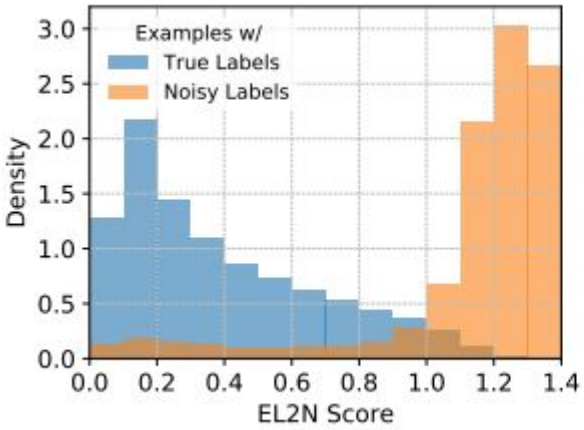
Low score - easy / typical examples

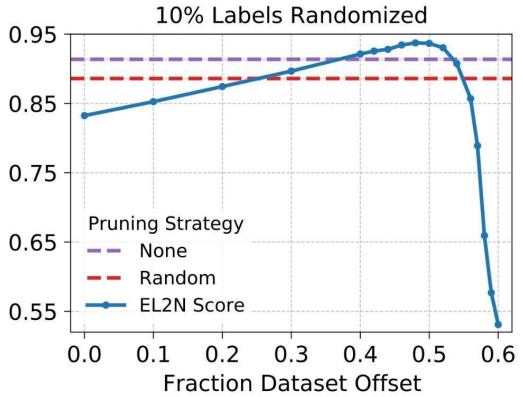


High score - difficult examples / outliers

Taking difficulty to the extreme

- Randomly select 10% of examples, and randomly re-assign a new wrong label;
- Trained network makes perfect predictions, indicating memorization.
- These random label examples get assigned high score.





Removing these high-score memorized examples improved generalization!

Key learnings from data diet

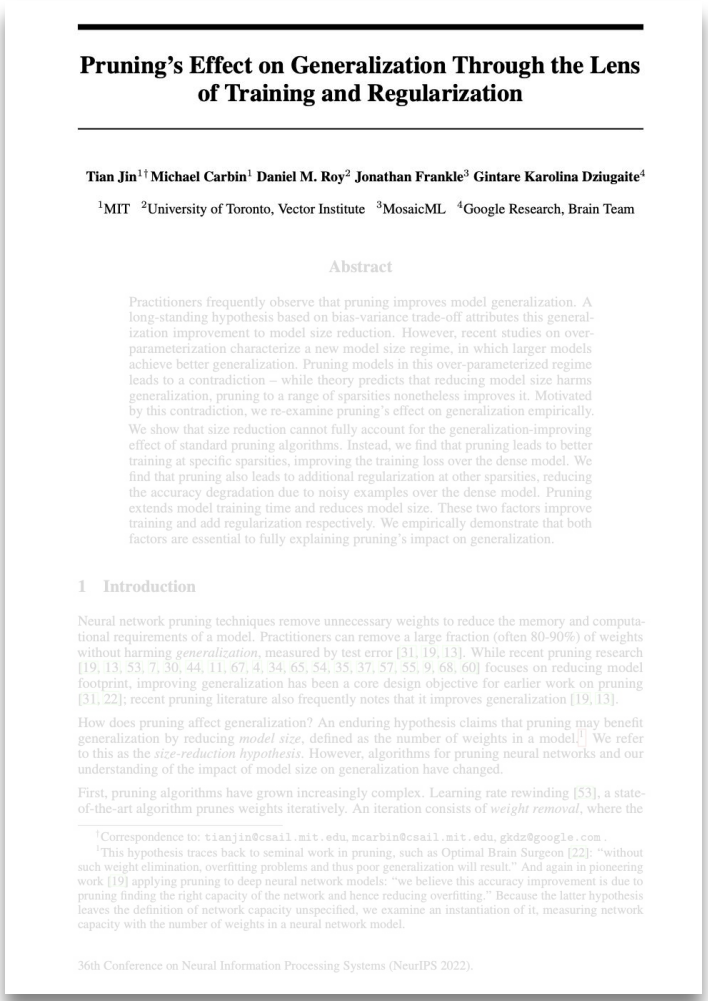
Most difficult examples tend to be:

- memorized and
- have high influence on generalization.

Influence: positive or negative?

Depends on the nature of the data.

If *negative*: removing these examples,
and thus avoiding their memorization
may *improve* generalization.



Pruning's Effect on Generalization Through the Lens of Training and Regularization

Tian Jin[1†] Michael Carbin[1] Daniel M. Roy[2] Jonathan Frankle[3] Gintare Karolina Dziugaite[4]

[1]MIT [2]University of Toronto, Vector Institute [3]MosaicML [4]Google Research, Brain Team

Abstract

Practitioners frequently observe that pruning improves model generalization. A long-standing hypothesis based on bias-variance trade-off attributes this generalization improvement to model size reduction. However, recent studies on over-parameterization characterize a new model size regime, in which larger models achieve better generalization. Pruning models in this over-parameterized regime leads to a contradiction – while theory predicts that reducing model size harms generalization, pruning to a range of sparsities nonetheless improves it. Motivated by this contradiction, we re-examine pruning's effect on generalization empirically. We show that size reduction cannot fully account for the generalization-improving effect of standard pruning algorithms. Instead, we find that pruning leads to better training at specific sparsities, improving the training loss over the dense model. We find that pruning also leads to additional regularization at other sparsities, reducing the accuracy degradation due to noisy examples over the dense model. Pruning extends model training time and reduces model size. These two factors improve training and add regularization respectively. We empirically demonstrate that both factors are essential to fully explaining pruning's impact on generalization.

1 Introduction

Neural network pruning techniques remove unnecessary weights to reduce the memory and computational requirements of a model. Practitioners can remove a large fraction (often 80-90%) of weights without harming *generalization*, measured by test error [31, 19, 13]. While recent pruning research [19, 13, 53, 7, 30, 44, 11, 67, 4, 34, 65, 54, 35, 37, 57, 55, 9, 68, 60] focuses on reducing model footprint, improving generalization has been a core design objective for earlier work on pruning [31, 22]; recent pruning literature also frequently notes that it improves generalization [19, 13].

How does pruning affect generalization? An enduring hypothesis claims that pruning may benefit generalization by reducing *model size*, defined as the number of weights in a model.[1] We refer to this as the *size-reduction hypothesis*. However, algorithms for pruning neural networks and our understanding of the impact of model size on generalization have changed.

First, pruning algorithms have grown increasingly complex. Learning rate rewinding [53], a state-of-the-art algorithm prunes weights iteratively. An iteration consists of *weight removal*, where the

[†]Correspondence to: tianjin@csail.mit.edu, mcarbin@csail.mit.edu, gkdz@google.com .

[1]This hypothesis traces back to seminal work in pruning, such as Optimal Brain Surgeon [22]: "without such weight elimination, overfitting problems and thus poor generalization will result." And again in pioneering work [19] applying pruning to deep neural network models: "we believe this accuracy improvement is due to pruning finding the right capacity of the network and hence reducing overfitting." Because the latter hypothesis leaves the definition of network capacity unspecified, we examine an instantiation of it, measuring network capacity with the number of weights in a neural network model.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Similar message from our pruning work:

Underfitting *some* examples may improve generalization

Some memorization is needed for good generalization...

(at least fixing other params, like SGD and its variants, architectures, data we deal with, etc...)

... and *not all* memorization is good.

Asking for no memorization may be too much.

Ask for less!

To what extent can we control memorization without hurting generalization?

0. Employ differential privacy: well-studied, simple adjustment to training, known limitations, effects on performance, etc. NEEDS TONS OF DATA.

1. Control unwanted memorization example frequency in the training set.
2. Apply unlearning to mitigate memorization.
3. Consider pruning - turns out facts are affected but not ICL! – can combine with external memory approaches
4. Modify architecture (through modularity, RAG, etc.)

Quantifying Memorization Across Neural Language Models

Nicholas Carlini[*] **Daphne Ippolito**[1,2] **Matthew Jagielski**[1]
Katherine Lee[1,3] **Florian Tramèr**[1] **Chiyuan Zhang**[1]

[1]*Google Research*
[2]*University of Pennsylvania*
[3]*Cornell University*

Abstract

Large language models (LMs) have been shown to memorize parts of their training data, and when prompted appropriately, they will emit the memorized training data verbatim. This is undesirable because memorization violates privacy (exposing user data), degrades utility (repeated easy-to-memorize text is often low quality), and hurts fairness (some texts are memorized over others).

We describe three log-linear relationships that quantify the degree to which LMs emit memorized training data. Memorization significantly grows as we increase (1) the capacity of a model, (2) the number of times an example has been duplicated, and (3) the number of tokens of context used to prompt the model. Surprisingly, we find the situation becomes more complicated when generalizing these results across model families. On the whole, we find that memorization in LMs is more prevalent than previously believed and will likely get worse as models continues to scale, at least without active mitigations.

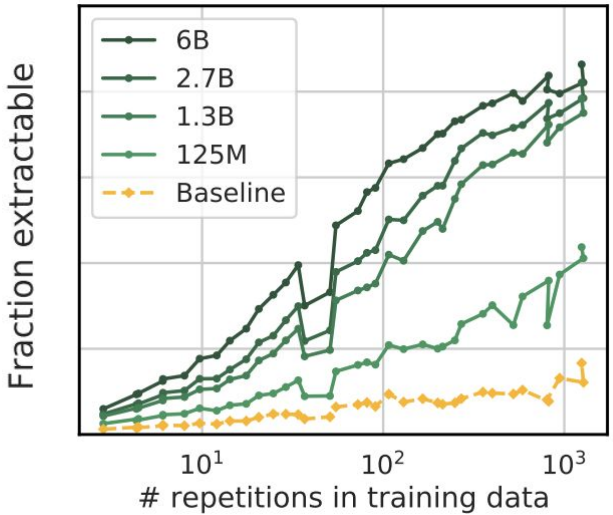
1 Introduction

The performance of neural language models has continuously improved as these models have grown from millions to trillions of parameters (Fedus et al., 2021), with their training sets similarly growing from millions to trillions of tokens. In anticipation of future, even larger models trained on minimally curated datasets, it is important to quantify factors that lead to increased memorization of a model's training set. Indeed, recent work has shown that *training data extraction attacks* are a practical threat for current language models (Carlini et al., 2020): an adversary interacting with a pretrained model can extract individual sequences that were used to train the model.

While current attacks are effective, they only represent a lower bound on how much memorization occurs in existing models. For example, by querying the GPT-2 language model, Carlini et al. (2020) (manually) identified just 600 memorized training examples out of a 40GB training dataset. This attack establishes a (loose) lower bound that at least 0.00000015% of the dataset is memorized. In contrast, we are able to show that the 6 billion parameter GPT-J model (Black et al., 2021; Wang and Komatsuzaki, 2021) **memorizes at least 1% of its training dataset**: The Pile (Gao et al., 2020).

In addition to prior work's loose estimates of models' memorization capabilities, there is a limited understanding of how memorization varies across different neural language models and datasets of different scales. Prior studies of memorization in language models either focus on models or datasets of a fixed size (Carlini et al., 2019; Zhang et al., 2021; Thakkar et al., 2020) or identify a narrow memorization-versus-scale relationship (Carlini et al., 2020; Lee et al., 2021). While McCoy et al. (2021) broadly study the extent to which language models memorize, their focus is on how to avoid the problem and ensure novelty of model outputs, rather than on studying model risk through identifying the maximal amount of data memorization.

[*] Authors ordered alphabetically.



Control unwanted memorization via controlling example frequency in the training set

To what extent can we control memorization without hurting generalization?

0. Employ differential privacy: well-studied, simple adjustment to training, known limitations, effects on performance, etc. NEEDS TONS OF DATA.

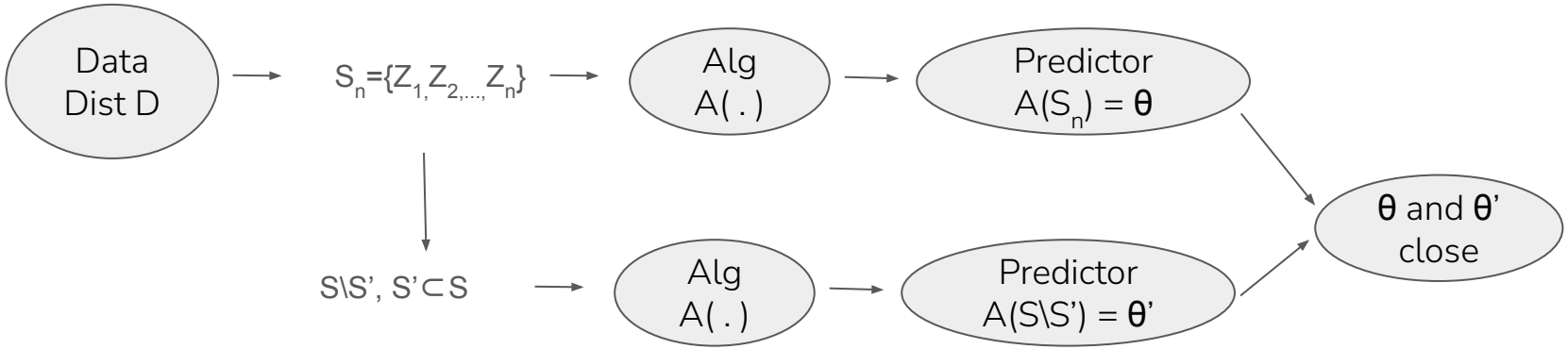
1. Control unwanted memorization example frequency in the training set.

2. Apply unlearning to mitigate memorization.

3. Consider pruning - turns out facts are affected but not ICL! – can combine with external memory approaches

4. Modify architecture (through modularity, RAG, etc.)

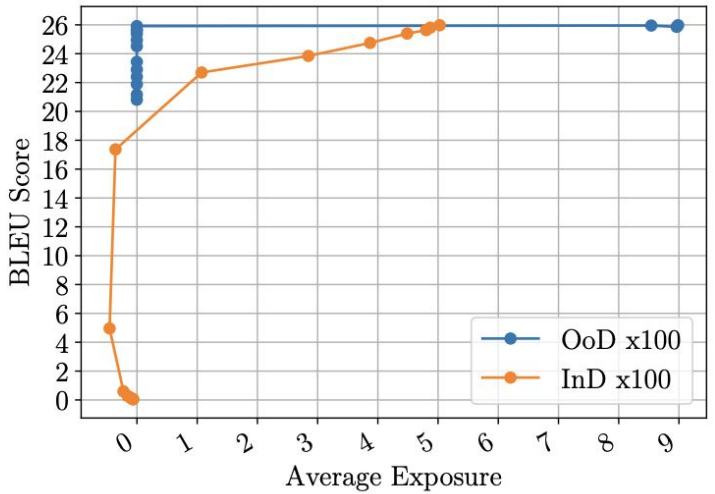
What's unlearning?



Unlearning is not equally effective for all examples

Compared unlearning and performance trade-off for in vs out of distribution samples:

- Unlearning memorized outliers is “easier” - does not damage performance.
- Unlearning in-distribution samples may be detrimental to performance.



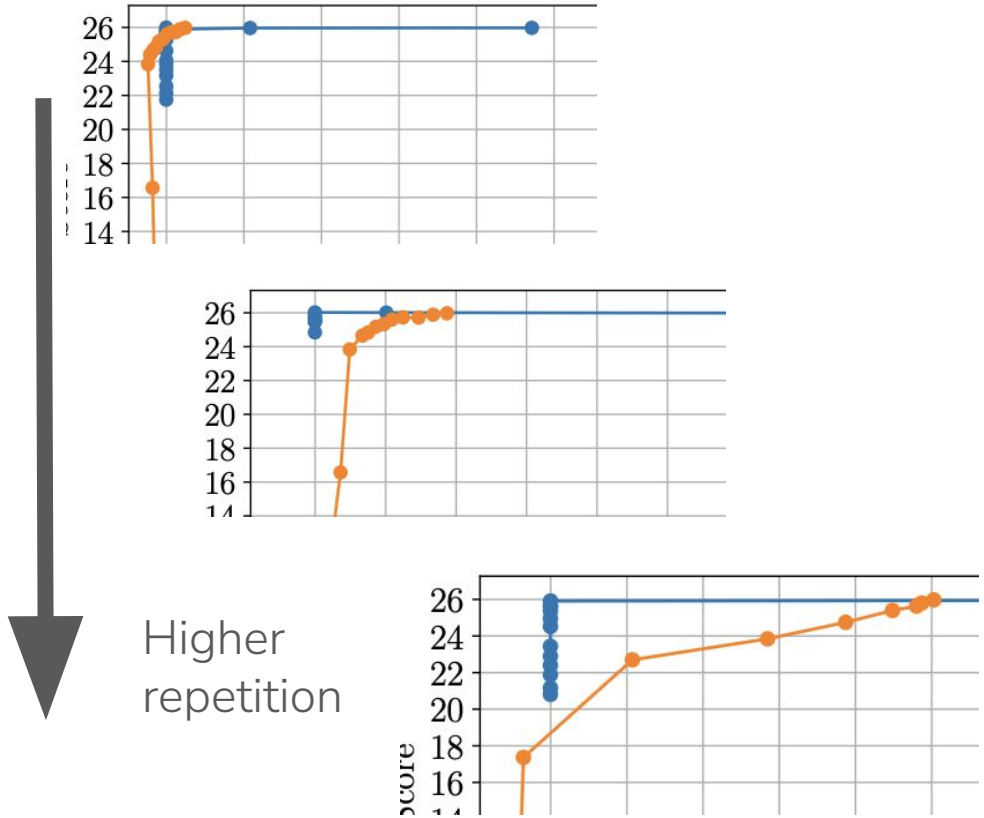
Exposure - how likely is it that the model generates the string of interest vs other “related” strings

How does repetition affect unlearning?

High repetitions

→ More difficult to unlearn

(at least for in-distribution examples)

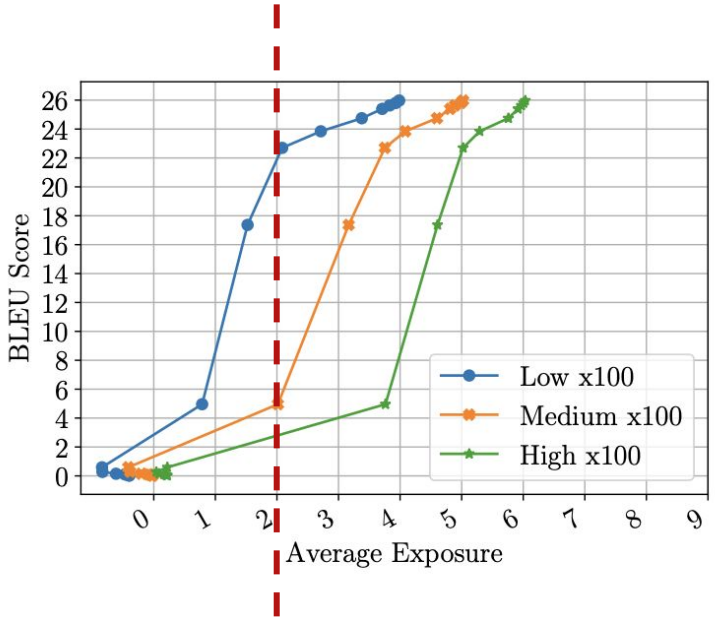


Connection between example difficulty and unlearning

More difficult examples

→ More difficult to unlearn

(differences are especially large under high repetitions)



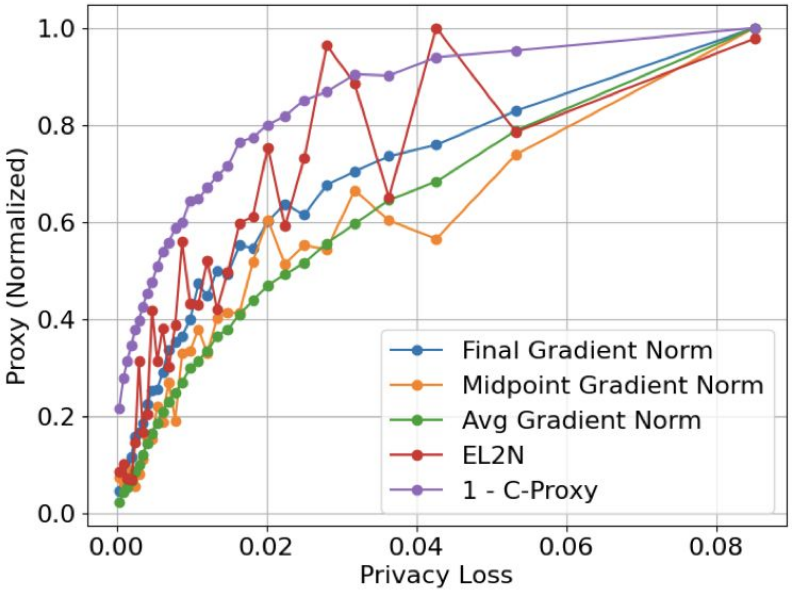
(a) $F^{\text{InD}}_{X\times 100}$.

Moving Beyond Heuristics: from data difficulty to privacy

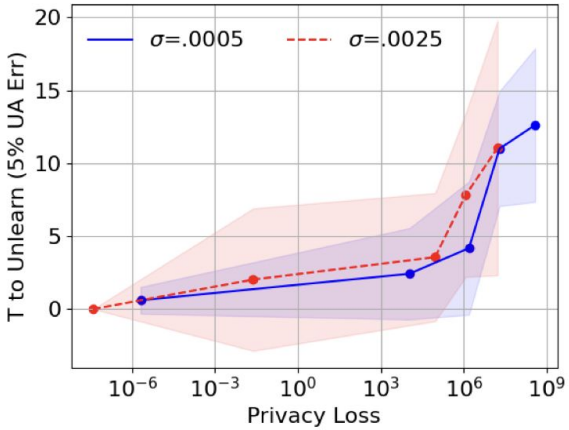
The Problem: Unlearning difficulty is data-dependent, but we lack a formal way to measure it.

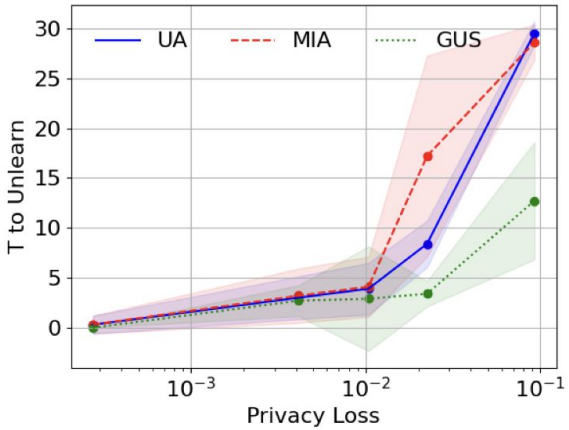
The Proposal: Leverage per-instance privacy loss:

Instead of a single, worst-case DP guarantee per-instance privacy loss bounds Renyi divergence between a model trained with and without a specific data point (Thudi et al.).



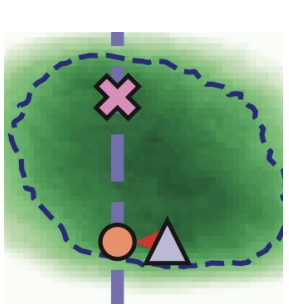
Per-Instance Privacy Loss Predicts Unlearning Difficulty

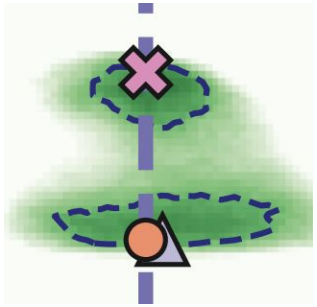


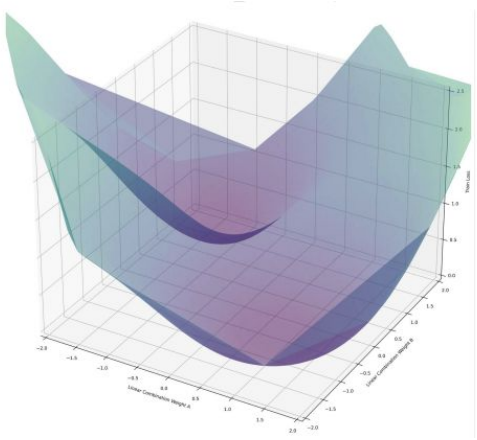


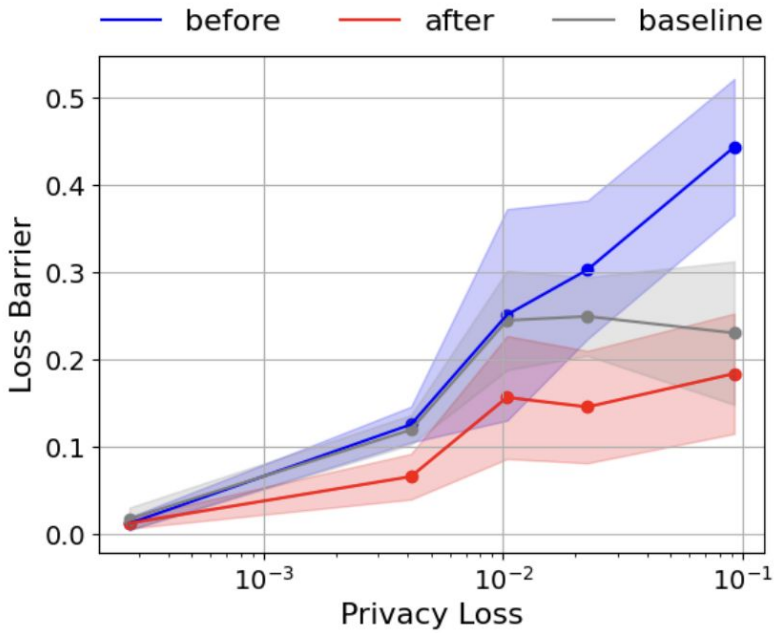
↕ per-instance privacy loss ⇒ ↕ more unlearning steps

Geometric intuition









Unlearning may mitigate memorization …

But with some caveats.

- Depends on the nature of the set we want to unlearn;
- How many times this data appeared during training;
- And what unlearning algorithm was used.
- Could other examples her “more memorized” as a result? (Privacy onion effect)

To what extent can we control memorization without hurting generalization?

0. Employ differential privacy: well-studied, simple adjustment to training, known limitations, effects on performance, etc. NEEDS TONS OF DATA.

1. Control unwanted memorization example frequency in the training set.

2. Apply unlearning to mitigate memorization.

3. Consider pruning - turns out facts are affected but not ICL! – can combine with external memory approaches

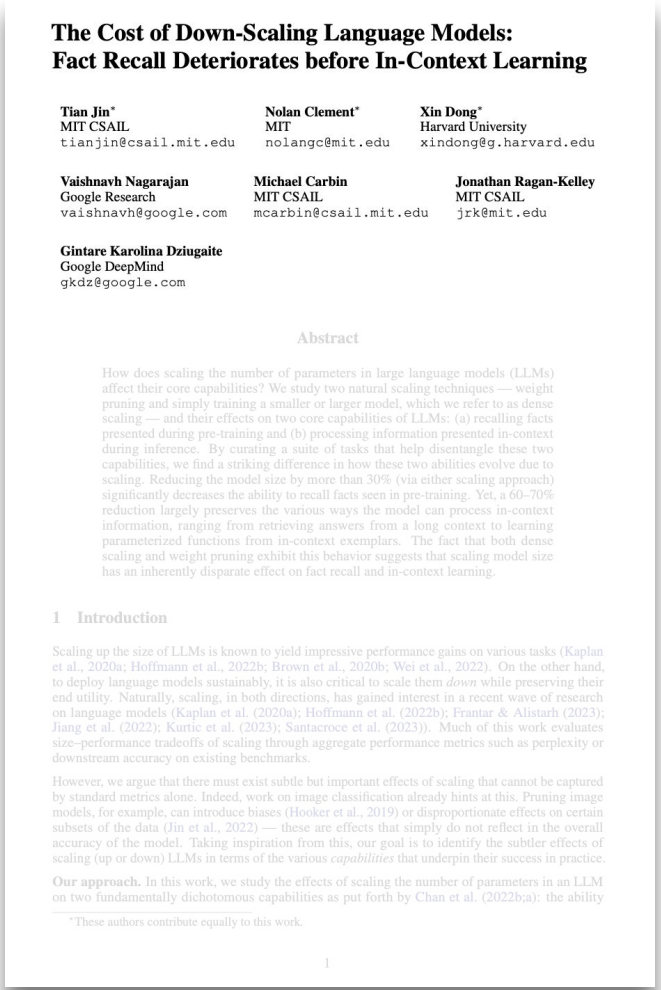
4. Modify architecture (through modularity, RAG, etc.)

Can pruning be used to mitigate memorization?

We've seen evidence that pruning introduces data-dependent regularization in classification.

How about in LLMs?

How does pruning affect different capabilities?



The Cost of Down-Scaling Language Models: Fact Recall Deteriorates before In-Context Learning

Tian Jin* MIT CSAIL tianjin@csail.mit.edu

Nolan Clement* MIT nolangc@mit.edu

Xin Dong* Harvard University xindong@g.harvard.edu

Vaishnavh Nagarajan Google Research vaishnavh@google.com

Michael Carbin MIT CSAIL mcarbin@csail.mit.edu

Jonathan Ragan-Kelley MIT CSAIL jrk@mit.edu

Gintare Karolina Dziugaite Google DeepMind gkdz@google.com

Abstract

How does scaling the number of parameters in large language models (LLMs) affect their core capabilities? We study two natural scaling techniques — weight pruning and simply training a smaller or larger model, which we refer to as dense scaling — and their effects on two core capabilities of LLMs: (a) recalling facts presented during pre-training and (b) processing information presented in-context during inference. By curating a suite of tasks that help disentangle these two capabilities, we find a striking difference in how these two abilities evolve due to scaling. Reducing the model size by more than 30% (via either scaling approach) significantly decreases the ability to recall facts seen in pre-training. Yet, a 60–70% reduction largely preserves the various ways the model can process in-context information, ranging from retrieving answers from a long context to learning parameterized functions from in-context exemplars. The fact that both dense scaling and weight pruning exhibit this behavior suggests that scaling model size has an inherently disparate effect on fact recall and in-context learning.

1 Introduction

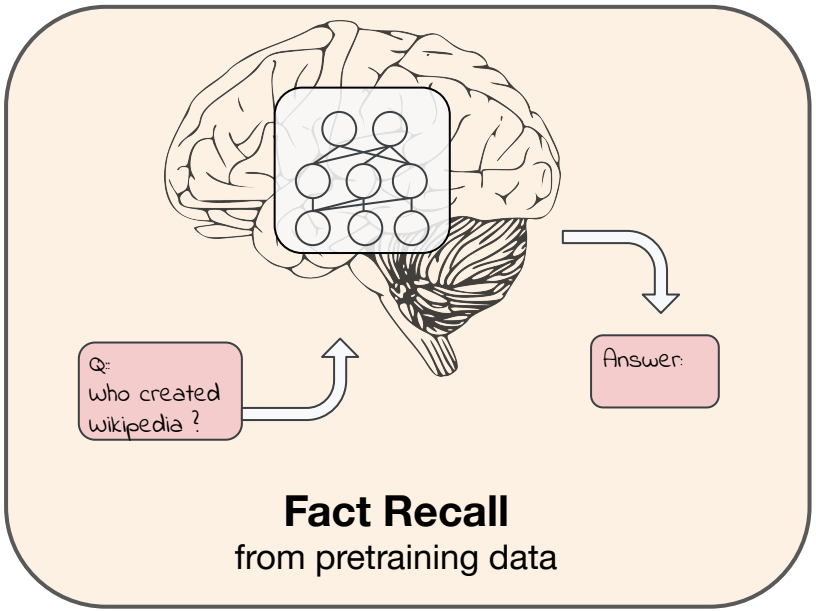
Scaling up the size of LLMs is known to yield impressive performance gains on various tasks (Kaplan et al., 2020a; Hoffmann et al., 2022b; Brown et al., 2020b; Wei et al., 2022). On the other hand, to deploy language models sustainably, it is also critical to scale them *down* while preserving their end utility. Naturally, scaling, in both directions, has gained interest in a recent wave of research on language models (Kaplan et al. (2020a); Hoffmann et al. (2022b); Frantar & Alistarh (2023); Jiang et al. (2022); Kurtic et al. (2023); Santacroce et al. (2023)). Much of this work evaluates size–performance tradeoffs of scaling through aggregate performance metrics such as perplexity or downstream accuracy on existing benchmarks.

However, we argue that there must exist subtle but important effects of scaling that cannot be captured by standard metrics alone. Indeed, work on image classification already hints at this. Pruning image models, for example, can introduce biases (Hooker et al., 2019) or disproportionate effects on certain subsets of the data (Jin et al., 2022) — these are effects that simply do not reflect in the overall accuracy of the model. Taking inspiration from this, our goal is to identify the subtler effects of scaling (up or down) LLMs in terms of the various *capabilities* that underpin their success in practice.

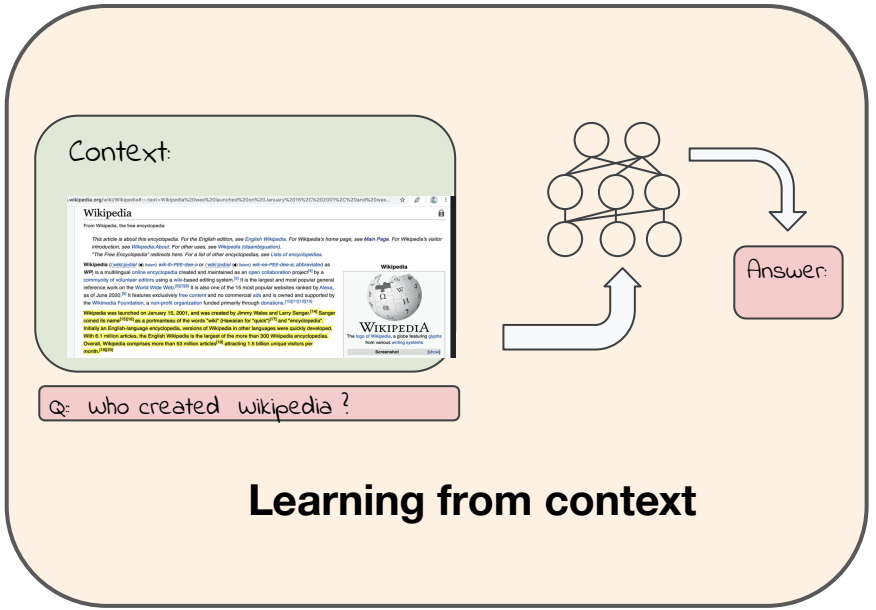
Our approach. In this work, we study the effects of scaling the number of parameters in an LLM on two fundamentally dichotomous capabilities as put forth by Chan et al. (2022b;a): the ability

*These authors contribute equally to this work.

Two key capabilities: Fact Recall vs ICL



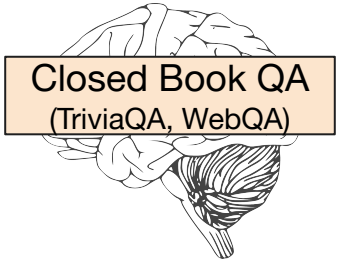
Fact Recall
from pretraining data



Learning from context

Most generic tasks demand both capabilities.
How do we disentangle them?

Disentangling fact recall and ICL capabilities



Closed Book QA
(TriviaQA, WebQA)

Question: what is...

≠

Overriding QA
(DissentQA)
(Neeman et al. 2022,
Li et al., 2022;
Longpre et al., 2021)

Context:

Question: what is ...

Fact Recall

In-context Learning

Open Book QA
(TriviaQA, NaturalQA)

Context:

Question: what is...

(*f* is
linear/
decision tree/
perceptron)

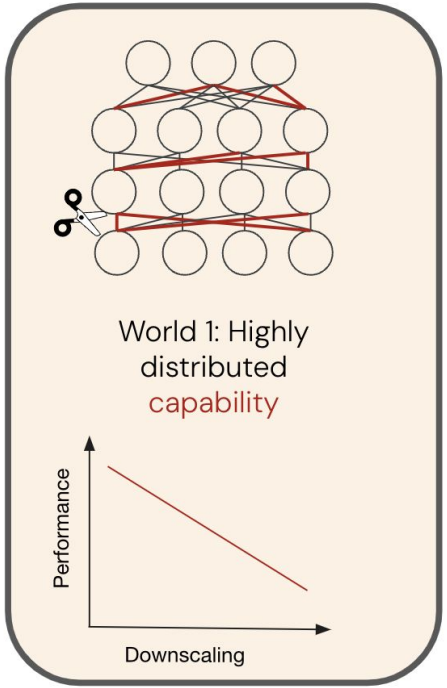
Generic Classification

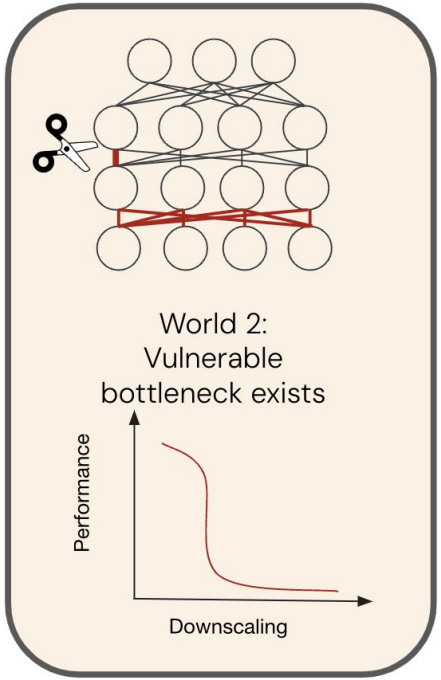
Context:

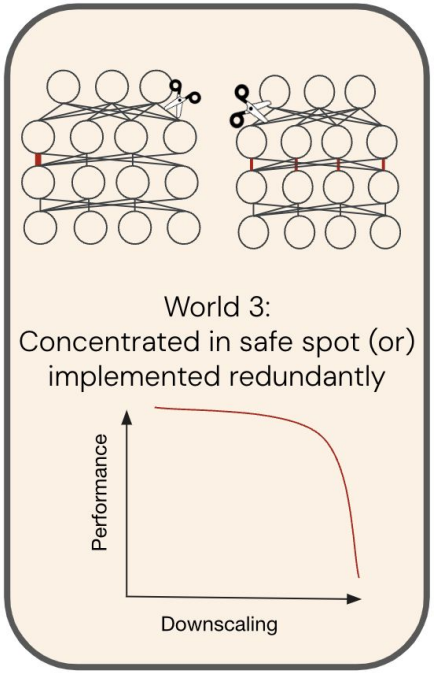
x_1, $f(x_1)$
x_2, $f(x_2)$
x_3, $f(x_3)$

Question: x_{test}

How could pruning affect a capability?



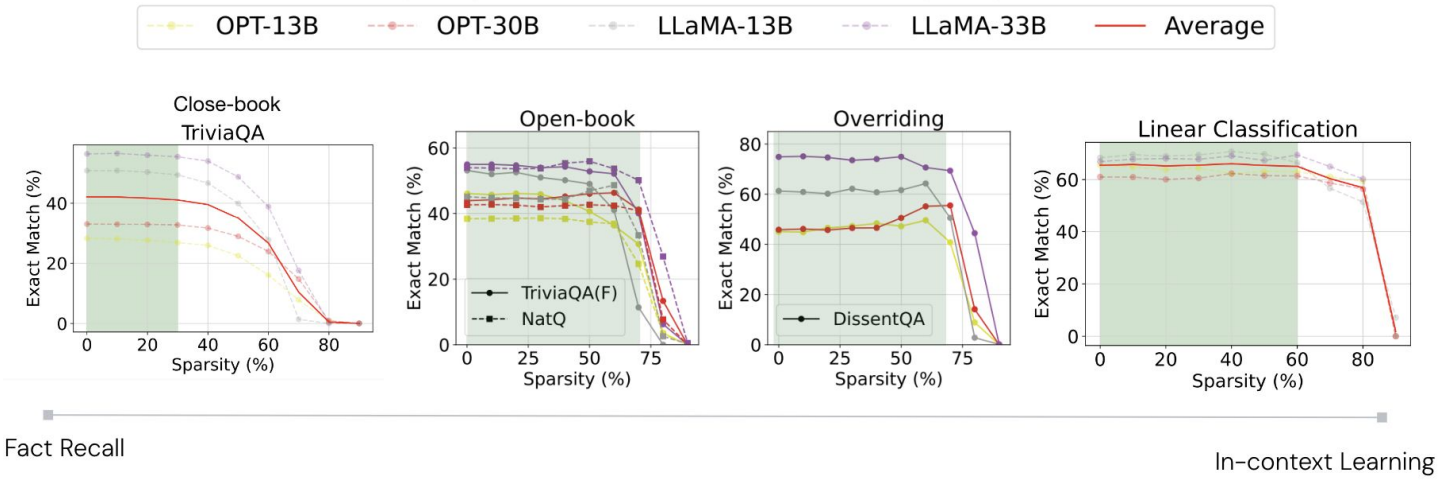




Pruning hurts fact recall first!

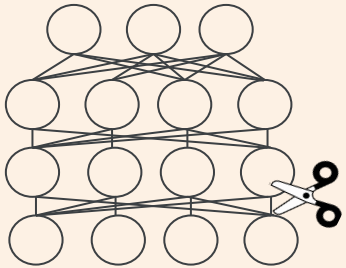
Fact recall deteriorates quicker (5% drop around 30% of pruning)

while in-context learning withstands as much as 60-70% pruning



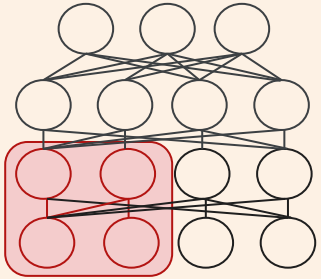
Pruning seems to be promising as a tool to mitigate fact memorization!

How can these findings be used?



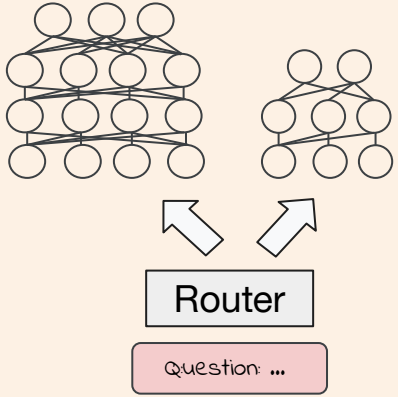
Improve pruning

Prune MLP layers more than attention?



Improve interpretability

Is there a small module that is responsible for in-context learning?



Control fact recall and Improve inference-time compute efficiency

Retrieve evidence into context + route to smaller model

To what extent can we control memorization without hurting generalization?

0. Employ differential privacy: well-studied, simple adjustment to training, known limitations, effects on performance, etc. NEEDS TONS OF DATA.

1. Control unwanted memorization example frequency in the training set.

2. Apply unlearning to mitigate memorization.

3. Consider pruning - turns out facts are affected but not ICL! – can combine with external memory approaches

4. Modify architecture (through modularity, RAG, etc.)

Summary

- Sometimes memorization is not only benign, it's *necessary*.

- Not all memorization is good for generalization!
  - Maybe somewhat mitigated by overparameterization. But costly!
  - Careful data selection, curation and curriculum matter;
  - May want to employ some form of data-dependent regularization.

- Removing memorization completely might not be a good choice.
  - Unless massive datasets (relative to D) are available for DP learning to actually work well. But as we scale the data, we scale D....

- Memorization can be mitigated in creative ways.
  - Unlearning; Pruning; Architectural changes (e.g., modularity, external memory)

Open Questions

Theoretical...

- What's the tradeoff between learning and memorization?
- Can we identify *what to memorize* to maximize generalization?
- What does the necessity to memorize mean for unlearning?

Empirical...

- What architectural changes would be best for controlling information access without losing performance?
- Can we identify ahead of time which examples are most likely to be memorized?