What do Learning Dynamics Reveal about Generalization in LLM Reasoning?

Anonymous Author(s)
Affiliation
Address
email

Abstract

When large language models (LLMs) are finetuned on reasoning tasks, they can either reduce their training loss by developing problem-solving abilities, or by simply memorizing target traces in the training data. Our work aims to better understand how this learning process shapes a model's ability to generalize. We observe that, while LLMs often perfectly memorize most target solution traces by the end of training, their predictions at intermediate checkpoints can provide valuable insights into their behavior at test time. Concretely, we introduce the concept of *pre-memorization train accuracy*: the accuracy of model samples for training queries prior to exactly reproducing reasoning traces in the training data. We find that the average pre-memorization train accuracy of the model is strongly predictive of its test performance, with coefficients of determination around or exceeding 0.9 across various models (Llama3-8B, Gemma2-9B), datasets (GSM8k, MATH), and training setups. Beyond this aggregate statistic, we find that the pre-memorization train accuracy of individual examples can predict the model's sensitivity to input perturbations for those examples, allowing us to identify examples for which the model fails to learn robust solutions. Our findings can offer guidance for training workflows, such as data curation, to improve generalization.

1 Introduction

Despite the remarkable capabilities of modern large language models (LLMs), the mechanisms behind their problem-solving abilities remain elusive. While some evidence suggest that models memorize vast amounts of data and reproduce similar patterns at test-time [Carlini et al., 2022, Bender et al., 2021], others find that LLMs develop complex problem-solving algorithms [Wang et al., 2022, Todd et al., 2023]. As an example, consider several LLMs finetuned on identical datasets and pretrained models, as illustrated in Fig. 1. While several models achieve near-perfect accuracy on the training data, their test performances differ significantly. This raises a crucial question: *What is the interplay between a model's learning dynamics and its ability to generalize to new problems?*

To investigate this question, we focus on mathematical reasoning tasks, where models are trained to generate both a final answer and intermediate reasoning steps. Although each problem has a single correct answer, the reasoning steps in a target solution trace represent just one of many valid ways to solve a problem. Therefore, a model that has memorized the training data is likely to replicate the exact reasoning steps from the solution trace for a problem in the training data, while a model with general problem-solving skills may produce the correct final answer but follow a different reasoning path. By analyzing model responses on training queries, focusing on both the accuracy of the final answer and the similarity of the response to the target solution trace, we can gain insights into whether the model is memorizing training data or developing problem-solving abilities.

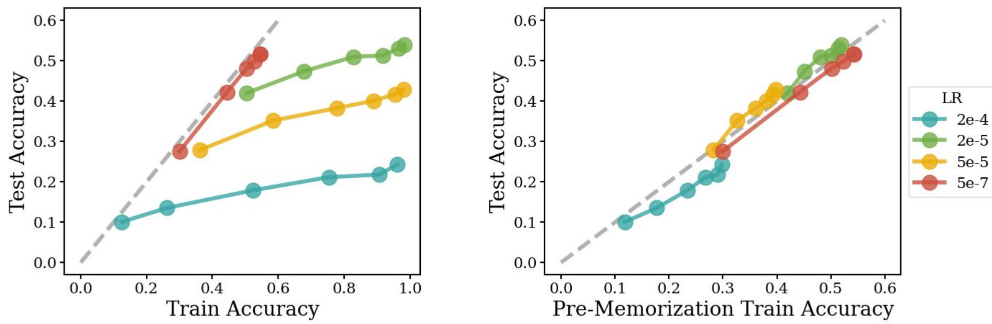

Figure 1: Relationship between train accuracy (left), pre-memorization train accuracy (right), and test accuracy for models finetuned on GSM8k using Llama3 8B. Each line represents a training run, and each point represents an intermediate checkpoint. Pre-memorization train accuracy strongly correlates with test accuracy, while train accuracy does not.

Our investigation reveals that analyzing model samples for training prompts at different stages of training can provide insight into the model's generalization behavior. While LLMs can often perfectly recall the entire finetuning dataset by the end of training, achieving perfect accuracy and exactly matching target solution traces, we observe distinct behaviors across training examples before full memorization occurs. For certain training examples, models only produce incorrect responses before memorizing the target response. For other examples, models first learn to generate diverse solution traces (distinct from the target solution trace) that all lead to the correct final answer, before later memorizing the target solution trace. Based on these observations, we introduce the concept of *pre-memorization train accuracy*, which refers to the highest accuracy a model achieves on a training example through the course of training before exactly memorizing the target solution trace. We find that a model's average pre-memorization train accuracy is highly predictive of the model's test accuracy, as illustrated in Fig. 1. Our experiments show that this phenomenon holds across different models (e.g., Llama3 8B [Dubey et al., 2024], Gemma2 9B [Team et al., 2024]), tasks (e.g., GSM8k [Cobbe et al., 2021], MATH [Hendrycks et al., 2021]), dataset sizes, and hyperparameter settings, with coefficients of determination around or exceeding 0.9.

Beyond predicting test accuracy, the pre-memorization accuracy of individual examples can provide insight into the robustness of model predictions for each example. To investigate the robustness of a model's prediction, we present the model with training queries accompanied by a short preamble, phrases like "First" or "We know that", that deviate from the target solution trace. While the model often performs nearly perfectly on unaltered training prompts, its accuracy tends to drop considerably for certain examples when faced with these modified prompts. We find that examples with low pre-memorization train accuracy are much more likely to show reduced performance. Thus, pre-memorization accuracy can help us to identify examples for which a model fails to learn robust solutions. This more granular view of model behavior not only allows us to ascertain how good we expect a model to be, but also determines the value of individual datapoints, providing actionable implications in training workflows, such as providing guidane for data curation.

2 Related Works

A number of works have studied the phenomenon of memorization during training, but consider different definitions of memorization. One definition quantifies memorization with "leave-one-out" performance, i.e., does the prediction of an example change significantly if we were to remove it from the training data? [Feldman and Zhang, 2020, Arpit et al., 2017, Zhang et al., 2021]. While we do not use this definition due to its computational cost, the notion of pre-memorization accuracy in our work captures a similar concept, allowing us to identify examples that a model fails to robustly learn without conducting expensive leave-one-out evaluations. In the context of language models, others have defined memorized examples as those where the model's output closely matches examples in the training data [Carlini et al., 2021, Tirumala et al., 2022, Inan et al., 2021, Hans et al., 2024], which has important privacy and copyright implications. Prior work has shown that this type of memorization is more likely to appear with duplicated data, larger model capacities, and longer context lengths [Carlini et al., 2022, Tirumala et al., 2022].

The relationship between the learning process and generalization has also been studied in a number of prior works. In particular, many works in this category focus on bounding the "generalization gap": the difference between training and test accuracies, using metrics related to model complexity, such as VC dimension or parameter norms [Neyshabur et al., 2015, Bartlett et al., 2019]. Other works focus on empirically motivated measures, such as gradient noise [Jiang et al., 2019] or distance of trained weights from initialization [Nagarajan and Kolter, 2019], to predict generalization. Jiang et al. [2019] conducted a comprehensive comparison of these methods and found that none consistently predicted generalization, though their work primarily focused on image classification. Other approaches have used unlabeled, held-out data to predict generalization, leveraging metrics such as the entropy of model predictions or the disagreement between different training runs [Garg et al., 2022, Platanios et al., 2016, Jiang et al., 2021]. Our findings suggest that pre-memorization accuracy can be a much stronger predictor of generalization in reasoning tasks with LLMs.

3 Preliminaries

In this work, we focus on training LLMs to perform reasoning tasks via finetuning. We are provided with a training dataset $D_{\text{train}} = \{(x_i, y_i)\}$, where queries x_i are drawn from $P(x)$ and solution traces y_i are drawn from $P(y|x)$. We assume the test dataset, D_{test}, is generated similarly to the training data. The model is finetuned by minimizing next-token prediction loss. We denote the finetuned model as $f_\theta(y|x)$, and model predictions as $\hat{y} \sim f_\theta(y|x)$.

In reasoning tasks, solution traces consist of both intermediate reasoning steps and a final answer, denoted as $\text{Ans}(y)$. The goal of reasoning tasks is for the model to generate solution traces with the correct final answer when faced with previously unseen queries. We measure the accuracy of model samples for a given query x_i using $\text{Acc}(f_\theta(y|x_i), y_i) = \mathbb{E}_{\hat{y}_i \sim f_\theta(y|x)}[1(\text{Ans}(\hat{y}_i) = \text{Ans}(y_i))]$. In our experiments, we approximate this accuracy by sampling from the model with a temperature of 0.8 and averaging the correctness attained by the samples.

While different solution traces drawn from $P(y|x)$ should all have the same final answer, they may contain different reasoning steps. Thus the target solution trace y_i of an example represents only one of many valid solution traces for solving x_i. We call an example *memorized* if the distance between the model's prediction and the target solution trace is low. Specifically, we consider $(x_i, y_i) \in D_{\text{train}}$ to be memorized by $f_\theta(y|x)$ if $\text{Perp}(f_\theta(y|x_i), y_i) < p$, where $\text{Perp}(f_\theta(y|x_i), y_i) = \exp(\frac{-1}{n_i} \log(f_\theta(y_i|x_i)))$, n_i is the number of tokens in y_i, and p is a threshold.

4 Characterizing the Learning Dynamics of LLM Reasoning Finetuning

In this section, we investigate the relationship between a model's learning progression during finetuning and its capacity for generalization. We begin by more precisely characterizing an LLM's learning process when finetuning on reasoning tasks. We focus on two key aspects of the model's behavior when presented with train queries: 1) whether the model's samples arrive at the correct final answer, and 2) the distance between the model's prediction and the target solution trace, measured by perplexity. These two metrics offer different perspectives on the model's behavior, because while there is only one correct final answer for each query, there may exist many different valid reasoning traces. Tracking both metrics through the course of training allows us to measure how effectively the model is able to solve training queries, and the extent to which this is accomplish by memorizing the target solution trace.

In Fig. 2, we visualize the learning progression, as characterized by the two metric described above, for three models finetuned on GSM8K. Each model is trained for three epochs, with a distinct peak learning rate that decays to zero by the end of training. As expected, training accuracy improves over time as the model minimizes the loss (color gradient from dark to light), and the distance between predictions and target solution traces decrease (from pink to yellow). For some learning rate settings, models approach near-perfect accuracy by the end of training, and their predictions closely match the target reasoning traces (mostly yellow in bottom row). However, during early stages of training, we observe significant differences in model behavior. For some examples, models initially produce incorrect predictions (black), and later replicate the target trace (yellow). For other examples, models first learn to generate correct answers with solution traces that differ from the target trace (pink), transitioning later to fully replicating the target trace (yellow).

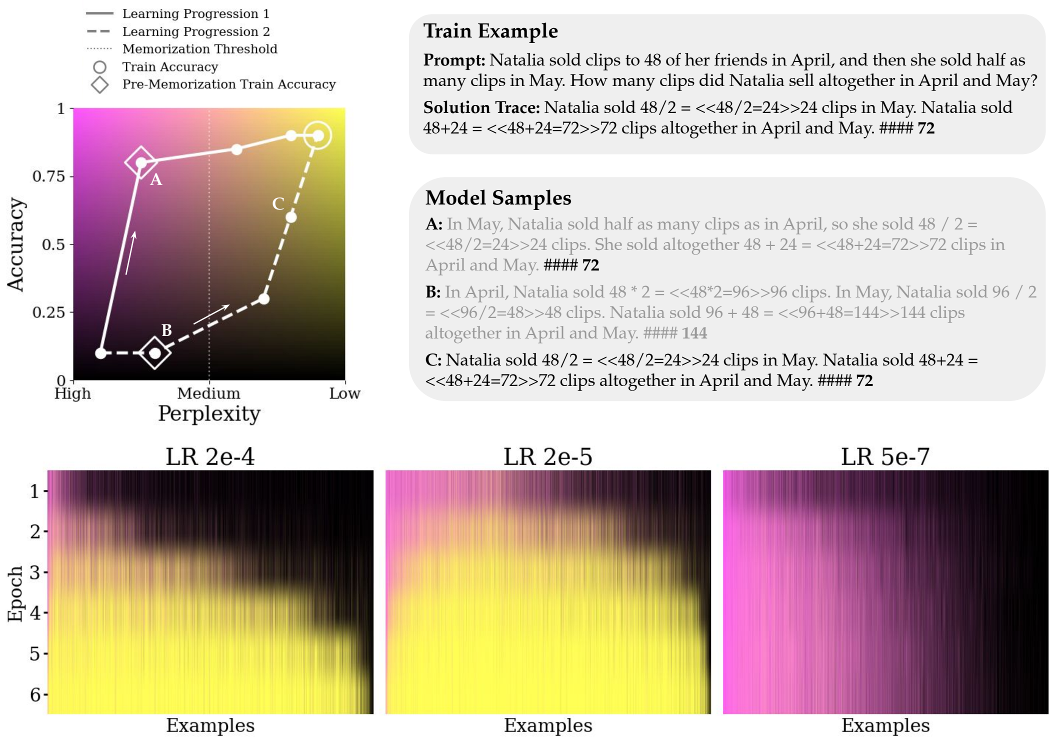

Figure 2: Visualizations of different learning progressions, as measured by the accuracy of model samples and the perplexity of target solution traces under model predictions. Top left presents a conceptual visualization, which represents accuracy with brightness and perplexity with color. Top right presents examples of model samples. Bottom plots visualize the predictions of 3 different models through the course of training. The x-axis represents individual training examples, the y-axis represents the epoch of training, and the color represents model predictions for each example in terms of accuracy and perplexity (legend in top left visualization).

We can see that with different training parameters, different models exhibit different capacities for generating accurate samples before memorizing target solution traces (amount of pink). This behavior suggests that the model has learned a problem-solving solution for the example, rather than a mapping to the exact target solution trace. We will more precisely study the connection between this behavior to test generalization in the next section, but first we introduce a new metric called pre-memorization accuracy to better quantify the accuracy of model samples before memorization.

We will use f_{θ_m} to denote the model at epoch m of training, with M as the total number of epochs. We first define a modified measure of accuracy as follows:

$$\text{Masked-Acc}(f_{\theta_m}(y|x_i), y_i, p) = \text{Acc}(f_{\theta_m}(y|x_i), y_i) \cdot 1[\text{Perp}(f_{\theta_m}(y|x_i), y_i) < p],$$

whose value is masked to zero if the model's prediction for that example is considered memorized (i.e., perplexity below p). We define the **pre-memorization accuracy** as follows:

$$\text{P-M Acc}(f_{\theta_{1:m}}(y|x_i), y_i, p) = \min\left\{\max_{1\leq m'\leq m} \text{Masked-Acc}(f_{\theta_{m'}}(y|x_i), y_i, p), \text{Acc}(f_{\theta_m}(y|x_i), y_i)\right\}$$

Unlike standard accuracy or masked accuracy, which evaluate performance at specific training checkpoints, pre-memorization accuracy evaluates the entire training process up to epoch m. We find that the average pre-memorization accuracy over the train data has a close relationship with the model's test accuracy. We will further elaborate on this relationship in the subsequent section.

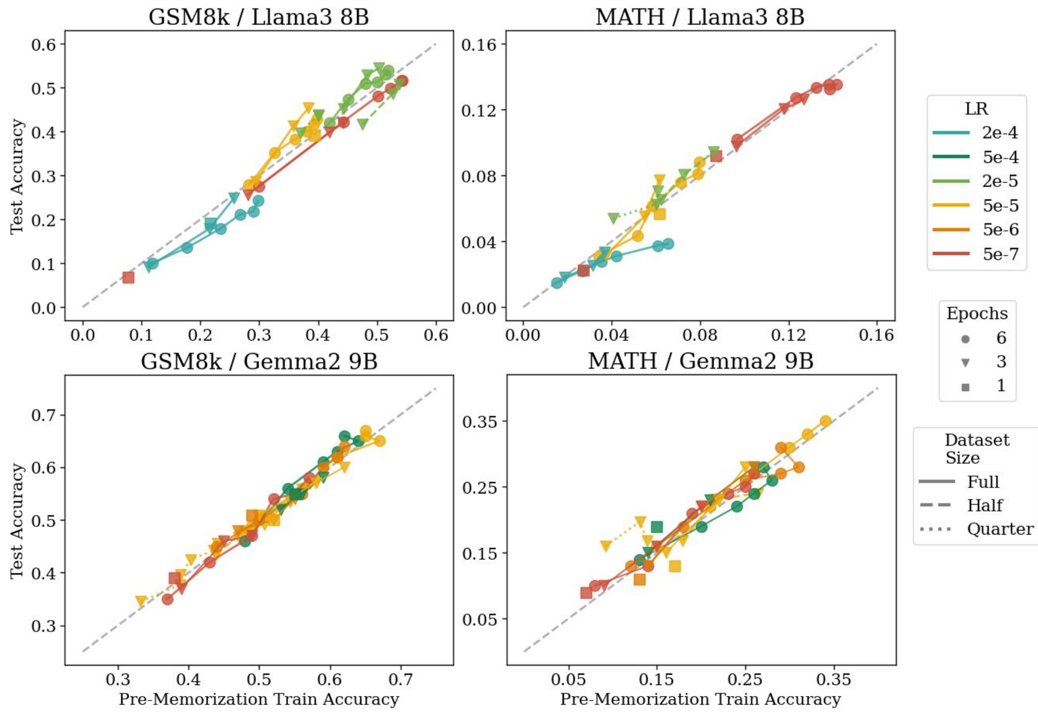

Figure 3: Evaluating the relationship between pre-memorization train accuracy and test accuracy. Each line corresponds to a training run, with each marker representing a specific checkpoint.

5 Pre-Memorization Train Accuracy Strongly Predicts Test Accuracy

As discussed in the previous section, pre-memorization train accuracy reflects the quality of the model's predictions before it begins to memorize the training data. Similarly, test accuracy captures the model's performance on unseen test examples that are never memorized (by construction, since they are never trained on). If the training and test distributions of problems match, then one may expect a model's pre-memorization accuracy to roughly reflect its test accuracy, since both quantities are evaluated on data from the same distribution and neither capture inflated accuracies due to memorization. Indeed, our experiments confirm this intuition. We find that **a model's average pre-memorization train accuracy is highly predictive of its test accuracy** across a variety of training runs and checkpoints. More concretely, we find that there exists a value of p for which a model's average pre-memorization train accuracy, $\mathbb{E}_{D_{\text{train}}}[\text{P-M Acc}(f_{\theta_{1:m}}(y|x_i), y_i, p)]$, closely approximates the model's test accuracy, $\mathbb{E}_{D_{\text{test}}}[\text{Acc}(f_{\theta_m}(y|x_i), y_i)]$. The value of p is dependent on the task and pretrained model, but not dependent on training parameters.

We illustrate this relationship in Fig. 3, where we plot the pre-memorization training accuracy and test accuracy across different training runs. We used Llama3 8B and Gemma2 9B as base models and GSM8K and MATH as the reasoning tasks. To evaluate different generalization behaviors, we finetuned the models by adjusting the peak learning rate (ranging from 5e-7 to 5e-4), the number of training epochs (1, 3, 6), and the dataset size (full, half, or quarter of the original dataset). We use the same value for p within each plot, and we find p by sweeping across a range of values. A full list of the training runs in our experiments and their training details can be found in Appendix C. We observe a strong linear relationship between pre-memorization training accuracy and test accuracy, with the results closely following the $y = x$ line across different models, tasks, and hyperparameter settings. More quantitatively, the coefficients of determination associated with each plot are 0.94 (GSM8k Llama), 0.95 (MATH Llama), 0.97 (GSM8k Gemma), and 0.88 (MATH Gemma). Our results show that pre-memorization training accuracy is a reliable predictor of test accuracy. We further compare pre-memorization train accuracy to several existing metrics for prediction generalization in Appendix A, and show our metric vastly outperforms prior metrics.

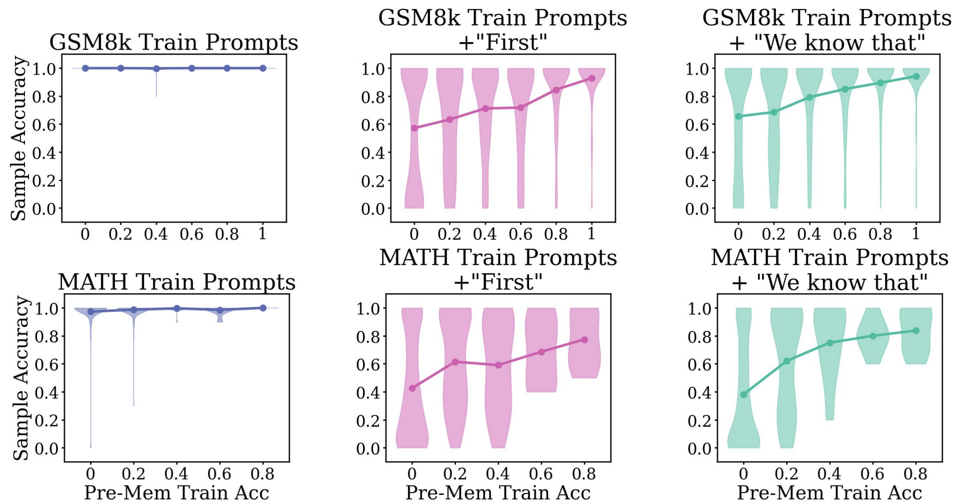

Figure 4: Accuracies of model samples (y-axis) when faced with the original prompt (left) and prompts with perturbations (middle, right). The x-axis represents the per-example pre-memorization train accuracy associated with each prompt. While the accuracy of model samples is almost perfect when faced with original prompts, it significantly degrades when faced with prompts with perturbations. Furthermore, the degradation of accuracy is much more significant for train examples with low pre-memorization accuracy than those with high pre-memorization accuracy.

6 Per-Example Pre-Memorization Accuracy Predicts Model Robustness

In the previous section, we demonstrated that a model's average pre-memorization accuracy provides insight into the model's overall generalization capability. In this section, we go beyond aggregate test accuracy and show that tracking per-example pre-memorization accuracy offers a window into the model's behavior at the level of individual training examples. We find that **model predictions tend to be less robust for train examples with low pre-memorization accuracy**.

To investigate the robustness of model predictions, we present the model with both the original training queries, as well as ones appended with short preambles to the solution trace, e.g. "First" or "We know that", which deviate from the target solution trace. In Fig. 4, we illustrate the prediction behavior of two models, both trained for six epochs with a learning rate of 2e-5, on the GSM8K and MATH datasets. We can see that while model performance is near-perfect for unaltered training prompts, certain examples exhibit a significant degradation in accuracy when presented with perturbed prompts. Furthermore, we can see that the accuracy of train examples with low pre-memorization train accuracy tend to degrate much more than those with high pre-memorization train accuracy.

Ideally, if the model has learned a robust problem-solving strategy, it should still be able to produce a valid reasoning trace, even with minor prompt deviations. Our findings suggest that pre-memorization train accuracy is a reliable indicator of the model's ability to generalize beyond memorization. This insight can be used to identify fragile examples where the model may have learned overly specific or non-generalizable patterns, which offers practical applications for improving model generalization. In Appendix B, we discuss how per-example pre-memorization accuracy can be used for data curation. Our experiments demonstrate that this approach leads to a 1.5-2x improvement in sample efficiency compared to i.i.d. data collection and outperforms other standard data curation techniques.

7 Conclusion

Our work introduces the concept of pre-memorization train accuracy, and shows that this metric is predictive of both test accuracy and per-element model robustness. These findings offer a deeper understanding of the relationship between a model's training dynamics and its ability to generalize.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pages 1183–1192. PMLR, 2017.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. *arXiv preprint arXiv:2201.04234*, 2022.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Abhimanyu Hans, Yuxin Wen, Neel Jain, John Kirchenbauer, Hamid Kazemi, Prajwal Singhania, Siddharth Singh, Gowthami Somepalli, Jonas Geiping, Abhinav Bhatele, et al. Be like a goldfish, don't memorize! mitigating memorization in generative llms. *arXiv preprint arXiv:2406.10209*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Huseyin A Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim. Training data leakage analysis in language models. *arXiv preprint arXiv:2101.05405*, 2021.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of sgd via disagreement. *arXiv preprint arXiv:2106.13799*, 2021.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning, 2024. URL `https://arxiv.org/abs/2312.15685`.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. Smaller language models are capable of selecting instruction-tuning training data for larger language models. *arXiv preprint arXiv:2402.10430*, 2024.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR, 2015.

Emmanouil Antonios Platanios, Avinava Dubey, and Tom Mitchell. Estimating accuracy from unlabeled data: A bayesian approach. In *International Conference on Machine Learning*, pages 1416–1425. PMLR, 2016.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Alex Tamkin, Dat Nguyen, Salil Deshpande, Jesse Mu, and Noah Goodman. Active learning helps pretrained models learn the intended task. *Advances in Neural Information Processing Systems*, 35:28140–28153, 2022.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Xueying Zhan, Qingzhong Wang, Kuan-hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B Chan. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450*, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L Zhang. A preliminary study of the intrinsic relationship between complexity and alignment. *arXiv preprint arXiv:2308.05696*, 2023.

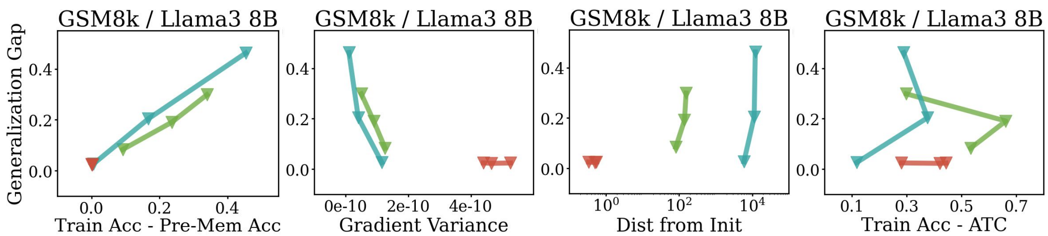

Figure 5: Evaluating different generalization metrics vs. the ground truth generalization gap for models finetuned on GSM8k using Llama3 8B (legend in Fig. 3). Our metric (left-most) is a much stronger predictor of the generalization gap than the other prior metrics.

A Comparison to Previous Generalization Metrics

As we discuss in Section 2, various metrics have been proposed in previous studies to predict the generalization gap, the difference between train and test accuracy. In Fig. 5, we compare several of these existing metrics, including gradient variance [Jiang et al., 2019], distance between current model weights and initialization [Nagarajan and Kolter, 2019], and an estimate of test accuracy via Average Thresholded Confidence (ATC) [Garg et al., 2022]. We discuss our implementation of these metrics in Appendix D. The correlation coefficients associated with each metric (left to right) are 0.98, -0.72, 0.59, -0.04, which shows that the prior metrics do not correlate as strongly with test accuracy as our proposed metric. A key advantage of our approach is that it leverages the assumption that model outputs include both reasoning steps and a final answer, enabling us to separate a model prediction's accuracy from its distance to the target solution trace. This distinction unveils the extent to which model's learn problem-solving solutions to training queries, which provides for a much more reliable estimate of a model's generalization abilities.

B Curating Data with Pre-Memorization Train Accuracy

We now present data curation as a practical application of the per-example understanding of generalization provided by pre-memorization train accuracy. While previous works have shown that prioritizing "hard" examples over "easy" ones during data collection can lead to more efficient scaling using heuristic measures of difficulty, the ideal metric for determining difficulty remains unclear. Our findings demonstrate that **pre-memorization accuracy can serve as a principled and more effective metric of example difficulty in data curation**.

Related Works A number of prior works have studied approaches for improving data curation. Active learning methods seek to optimally select data in an online learning setting for general machine learning models [Zhan et al., 2022, Gal et al., 2017, Tamkin et al., 2022]. Specific to LLM finetuning, prior approaches largely fall into three categories: optimization-based, model-based, and heuristic-based approaches. Optimization-based methods frame data selection as an optimization problem, where the objective is model performance, and the search space consists of the training data distribution [Engstrom et al., 2024, Grosse et al., 2023]. Model-based approaches, on the other hand, leverage characteristics of the learning process [Mekala et al., 2024, Liu et al., 2024], such as comparing the perplexity of examples [Li et al., 2023]. Lastly, heuristic-based methods rely on simpler criteria, such as difficulty scores generated by GPT, to classify desirable training data [Chen et al., 2023, Lu et al., 2023, Zhao et al., 2023]. Our data curation approach aligns most closely with model-based strategies, as we use the model's pre-memorization accuracy, a characteristic of the learning process, to inform the selection of training examples.

Problem setup. We will now more precisely define our data curation problem. Given an existing set of N training examples with queries distributed as $P(x)$, we aim to collect N' examples, denoted as D'_{train}, to augment the dataset. The goal is to specify a new distribution $P'(x)$ that maximizes the test performance of a model trained on both the original and the newly collected examples. While defining the true distribution of queries can be challenging, we assume that by approximating it with an empirical distribution from the current dataset, we can collect new data with similar properties. In our experiments, we take D_{train} to be the original dataset, and collect new examples by using GPT to

Algorithm 1 Our Data Collection Process

Input: $N' = N'_1 + \cdots + N'_n$, t
Output: Updated dataset D'_{train}
Initialize $D'_{\text{train}} = \{\}$
for $i = 1$ to n **do**
    Train model on $D_{\text{train}} + D'_{\text{train}}$
    Evaluate model on D_{train} and compute pre-memorization accuracy for each example
    Set $P'_i(x)$ as the distribution of examples with pre-memorization accuracy below t
    Collect N'_i new examples from $P'_i(x)$ and add them to D'_{train}
end for

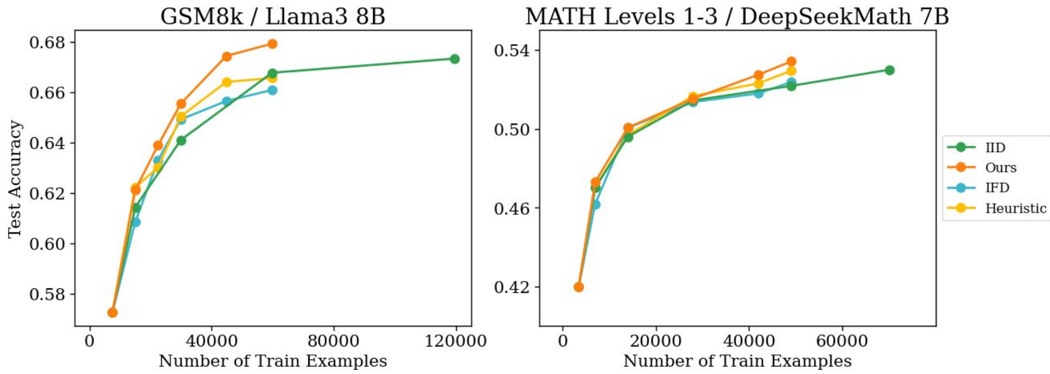

Figure 6: Comparison of different approaches for data curation. Each line represents a different data curation approach at varying scales of training dataset size, and each point represents a different training run. Our approach acheived the best sample efficiency compared to the other approaches.

rephrase examples in the original dataset, similar to the procedure in Setlur et al. [2024]. By only collecting new examples that derive from the specified empirical distribution, we can ensure the new dataset approximates $P'(x)$. This setup can also be used when collecting new human-generated data, by providing the specified empirical distribution of examples as references for human labelers.

Our approach. We propose a data collection strategy that focuses on examples with low pre-memorization accuracy in the existing dataset. First, we calculate the pre-memorization accuracy for each example in the current dataset and then define $P'(x)$ as the distribution of examples whose pre-memorization accuracy falls below a certain threshold t. We then collect new data according to this distribution. If N' is large, we can split the data collection process into multiple iterations ($N'_1 + ... + N'_n = N'$). In each iteration, we collect N'_i new examples according to $P'_i(x)$, retrain a model on the combined dataset, calculate the pre-memorization accuracy with the model, and update $P'_{i+1}(x)$ for the next round of data collection. This process is summarized in Algorithm 1.

Comparisons. We compare our strategy to IID sampling and two existing approaches commonly used in data curation. Both of these approaches propose a metric of example difficulty and prioritize difficult examples during data collection. The first metric, called Instruction-Following Difficulty (IFD) [Li et al., 2023], computes the ratio between the perplexity of training labels given inputs and the perplexity of only labels using a model finetuned for the task. The second metric uses external sources, such humans or more capable models such as GPT, to assign a heuristic notion of difficulty to each example [Chen et al., 2023, Lu et al., 2023, Zhao et al., 2023]. For the GSM8K dataset, we use the number of lines in the target solution traces as a heuristic for difficulty, while for the MATH dataset, we use the difficulty levels provided in the dataset itself.

Results. In our experiments, we finetune Llama3 8B on GSM8K and DeepSeekMath 7B [Shao et al., 2024] on MATH levels 1-3 to evaluate the impact of our data collection approach on test accuracy. More details about the implementation of the different approaches can be found in Appendix E. As shown in Fig. 6, our approach outperforms all three prior approaches, achieving more than 2x the sample efficiency in reaching the same test accuracy compared to IID scaling in GSM8k, and more than 1.5x sample efficiency in MATH levels 1-3. These results highlight the effectiveness of using pre-memorization accuracy as a criterion for targeted data collection, leading to enhanced generalization with fewer data points.

C Section 5 Training Runs Details

In this section, we will enumerate all training runs shown in Fig. 3 and their training details. For our half and quarter training runs, we fix the total number of training steps to be equivalent to training for 3 epochs on the full dataset.

C.1 GSM8k LLama3 8B

For all training runs with GSM8k and Llama3 8B, we use the AdamW optimizer, with a linear decay learning rate scheduler with 20 warmup steps, a batch size of 128, and a max gradient norm of 2.

Learning Rate	Epochs	Dataset Size
5e-5	6	full
2e-5	6	full
5e-7	6	full
2e-4	6	full
5e-5	3	full
2e-5	3	full
5e-7	3	full
2e-4	3	full
5e-5	1	full
5e-7	1	full
2e-4	1	full
2e-5	6	half
2e-5	12	quarter

C.2 MATH LLama3 8B

For all training runs with MATH and Llama3 8B, we use the AdamW optimizer, with a linear decay learning rate scheduler with 20 warmup steps, a batch size of 24, and a max gradient norm of 2.

Learning Rate	Epochs	Dataset Size
5e-5	6	full
5e-7	6	full
2e-4	6	full
5e-5	3	full
5e-7	3	full
2e-4	3	full
5e-5	1	full
5e-7	1	full
2e-4	1	full
2e-5	6	half
2e-5	12	quarter

C.3 GSM8k Gemma2 9B

For all training runs with GSM8k and Gemma2 9B, we use the Adam optimizer, with a cosine decay learning rate scheduler with (0.1*total steps) warmup steps, a batch size of 32, and a max gradient norm of 1.

Learning Rate	Epochs	Dataset Size
5e-4	6	full
5e-5	6	full
5e-6	6	full
5e-7	6	full
5e-4	3	full
5e-5	3	full
5e-6	3	full
5e-7	3	full
5e-4	1	full
5e-5	1	full
5e-6	1	full
5e-7	1	full
5e-5	6	half
5e-5	12	quarter

C.4 MATH Gemma2 9B

For all training runs with MATH and Gemma2 9B, we use the Adam optimizer, with a cosine decay learning rate scheduler with (0.1*total steps) warmup steps, a batch size of 32, and a max gradient norm of 1.

Learning Rate	Epochs	Dataset Size
5e-4	6	full
5e-5	6	full
5e-6	6	full
5e-7	6	full
5e-4	3	full
5e-5	3	full
5e-6	3	full
5e-7	3	full
5e-4	1	full
5e-5	1	full
5e-6	1	full
5e-7	1	full
5e-5	6	half
5e-5	12	quarter

D Appendix A Implementation Details

In this section we will more precisely describe each generalization metric.

D.1 Gradient Variance

We calculate the gradient of the model for 5 different minibatches, take the variance across the 5 samples for each element of each weight matrix, and take the average over each element of the model weights.

D.2 Distance from Initialization

We calculate the squared difference between each element of the model weights at initialization and after finetuning, and take the sun across all elements.

D.3 Average Thresholded Confidence (ATC)

ATC computes a threshold on a score computed on model confidence such that the fraction of examples above the threshold matches the test accuracy. For the score, we use the likelihood of greedily sampled responses under the model. We calculate the the score over the training data using a model trained for 3 epochs using learning rate 2e-5, and calculate the threshold over the score using the test dataset. We then predict the test accuracies over different models in our experiment by calculating the score associated with the training data using each model, and measuring the percentage of examples whose score surpass the threshold that we previously calculated.

E Appendix B Implementation Details

For our approach for data curation, we implemented the process described in Algorithm 1, with 5 iterations (n) and using threshold (t) 0.75 for both GSM8k and MATH.

For the IFD approach for data curation, we calculated the IFD score using a model that was train on the test set associated each dataset for 2 epochs. This is because, in order to calculated the IFD score, we need a model which has been briefly trained for the task of interest, but which has not been exposed to the dataset for which we want to calculate the IFD score over. Note that this model is only used for calculating for the IFD score, and not used for evaluations in our experiments, so there is no data leakage.

For both the IFD approach and the heuristic approach, we take $P'(x)$ to be top 50 percentile of examples for GSM8k, and top 75 percentile of examples for MATH. We designed these percentiles to roughly match the percentile of examples that our approach selects from.

For all training runs, we use the AdamW optimizer, with a linear decay learning rate scheduler with 20 warmup steps, a batch size of 128, a max gradient norm of 2, a learning rate of 2e-5, and 3 epochs of training.