Emergent and Predictable Memorization in Large Language Models

Stella Biderman
EleutherAI
Booz Allen Hamilton
stella@eleuther.ai

USVSN Sai Prashanth
EleutherAI
usai@eleuther.ai

Lintang Sutawika
EleutherAI
lintang@eleuther.ai

Hailey Schoelkopf
EleutherAI
Yale University
hailey@eleuther.ai

Quentin Anthony
EleutherAI
Ohio State University
quentin@eleuther.ai

Shivanshu Purohit
Stability AI
EleutherAI
shivanshu@stability.ai

Edward Raff
Booz Allen Hamilton
raff_edward@bah.com

ABSTRACT

Memorization, or the tendency of large language models (LLMs) to output entire sequences from their training data verbatim, is a key concern for safely deploying language models. In particular, it is vital to minimize a model's memorization of sensitive datapoints such as those containing personal identifiable information (PII). The prevalence of such undesirable memorization can pose issues for model trainers, and may even require discarding an otherwise functional model. We therefore seek to predict which sequences will be memorized before a large model's full train-time by extrapolating the memorization behavior of lower-compute trial runs. We measure memorization of the Pythia model suite, and find that intermediate checkpoints are better predictors of a model's memorization behavior than smaller fully-trained models. We additionally provide further novel discoveries on the distribution of memorization scores across models and data.

1 INTRODUCTION

Recent natural language processing (NLP) research in generative tasks has largely been driven by two findings: (1) The transformer architecture performs well [17, 42, 52]; and (2) Increasing the scale of transformer architectures leads to improved performance [8, 15]. In addition to these benefits, transformers are a general and multipurpose architecture that have achieved state-of-the-art results outside of NLP on diverse tasks such as text-to-image synthesis [16, 44, 45], code generation [14, 19, 55], and protein modeling [2, 30]. Despite their widespread success and increasing use, the internal workings of transformer models are poorly understood and research into how a given model learns and internally represents data has the potential to affect a broad range of high-impact applications.

1.1 Memorization in Large Language Models

In particular, the demonstrated capacity and ability of these large language models to memorize data has become a significant concern [11, 12, 27]. The most obvious ramification is personal information or otherwise sensitive data being leaked to the public at large and extracted by a bad actor. Although it has not been formally demonstrated in the literature (to the best of our knowledge), some forms of memorization are actually beneficial: we want large language models to memorize factual events and details to avoid "hallucinating" plausible-sounding but errant facts to unsuspecting users [10, 40, 50].

Despite the extensive literature on memorization in trained models [11, 12, 27], there are few tools to help practitioners either prevent memorization or detect it early in model training. Before the advent of transformer-based large language models work using differential privacy was popular [1, 35, 39]. However such methods have been observed to hurt performance during pretraining [3], and are therefore not popular among people who train large language models. In recent years, the bulk of interventionist work has focused on how removing duplicated samples from the training dataset (known as deduplication) can decrease memorization [13, 25, 31, 34]. Importantly, these works focus on memorization *on average* and cannot be relied on to prevent memorization of specific training examples. Ippolito et al. [28] introduce an interference-time intervention that has a 100% success rate at preventing *verbatim* memorization, but note that their methodology is easily subverted, and does not fulfill the intention behind the term "memorization," which is the tendency of models to learn entire samples *during training* without understanding their underlying meaning.

DP hurts performance

Both of these memorization outcomes can be better studied, tackled, and remediated if there exist tools to predict the memorization of *specific data points* prior to model training, rather than the macro-level corpus-wide statistics considered by prior work. We take a first step in this direction by proposing two strategies: 1) making predictions from a smaller model to a larger model; and 2) making predictions from a partially trained model of a given size to the fully trained model. Using smaller/partial model training runs is critical, as we want a method that is financially cheaper to run on a corpus than training an entire model from scratch; otherwise, one would simply train the full model and know for certain which data is memorized. We find that each proposed method works best in a different scenario, depending on whether precision (we want to

Small model → Large model

Partially trained model → Fully trained model confirm something was memorized) or recall (we want something "forgotten") is most desired. We believe that

1.2 Scaling Laws and Emergent Properties

Our approach to predicting memorization of specific sequences is inspired by the literature on scaling laws for large language models. Due to the substantial cost of training large language models, it is highly desirable to be able to make predictions about model characteristics before they are actually trained. The literature on *scaling laws* [23, 24, 26, 32, 37] has been successfully used to inform the decision-making of a variety of researchers at model training-time by allowing them to generalize the decisions made while investigating smaller models to inform the design of larger (sometimes by many orders of magnitude) models [7, 15, 43, 47]. While this work on scaling laws does extend to memorization [13, 25], how memorization evolves during a model's training process across a variety of scales has not been studied.

More recently, attention has been directed to areas where scaling laws (or, at least, traditional conceptions of them) fail [9, 49]. In particular, Ganguli et al. [20], Wei et al. [54], and Srivastava et al. [49] study what are termed "emergent properties" of language models, where some downstream tasks see almost no change in performance as the model size scales until a critical point at which performance increases rapidly. While these emergent properties can be beneficial, their existence can also imply that it is difficult to extrapolate findings from small models to larger models.

1.3 Our Contribution

In this paper we introduce the question of extrapolating a model's memorization behavior for *specific training data points* based on evaluations set in relatively low-cost training regimes. These low-cost regimes enable us to abort models without wasting significant compute resources in the event of undesirable behavior. This includes the *typical setting*, where we extrapolate the qualities of large models based on small models, as well as a *novel setting* where we extrapolate the behavior of fully-trained models based on partially-trained models. As far as we are aware, we are the first paper to study forecasting model behavior in this novel setting.

Our primary contributions are:

(1) Introducing the difficulties of both forecasting a future model's memorization of specific training data, and forecasting the memorization behavior of fully-trained LLMs from partially-trained checkpoints.
(2) The discovery that which data points are memorized by small-scale LLMs **do not** reliably generalize to larger-scale LLMs.
(3) The discovery that which data points are memorized by partially-trained LLMs **does** reliably generalize to fully-trained LLMs.

The rest of this paper is organized as follows: in Section 2 we present relevant facets of our methodology, including definitions of metrics (Sections 2.1 and 2.3), the threat model (Section 2.2), and choice of pretrained models (Section 2.4). We then investigate the broader distribution of strings that are only partially memorized in Section 6. In Section 3, we explore the feasibility of predicting the memorization behavior of large models based on small models. Further, in Section 4.1, we explore the feasibility of predicting the memorization behavior of the fully-trained model based on intermediate checkpoints. Finally, we measure how a model's memorization behavior is affected by duplicate data in Section 4.2.

2 METHODOLOGY

2.1 Measuring Memorization

"Memorization" is an intuitive concept that, for many people, stands distinct from "good learning" in some senses. However, formalizing this intuition presents challenges. In this paper, we consider the framework introduced by Carlini et al. [12] grounded in *k-elicitation*:

Definition 2.1. A string s is said to be k-elicitable if it (a) exists in the training data, and (b) is generated by the language model by prompting with k prior tokens.

To demonstrate, the training data sequence "*Their email address is me@alice.com*" is 3-elicitable (memorized) if the prompt "*Their email address*" yields "*is me@alice.com*"—thus producing an exact copy of the training data sequence. We term the number of tokens in the continuation the *memorization score* of the sequence and call a sequence (k-)memorized if the memorization score is 1. Illustrative examples are shown in Table 1

Doing a forward pass on a large transformer is relatively expensive, costing about one third the cost of a full gradient update step. Consequently, feeding the full training data through the model for a forward pass would cost approximately one third the amount of compute that training the model did, and doing the full seven checkpoints that we do would come out to a larger compute budget than training the models themselves.

To ensure computational feasibility in our experiments, we choose $k = 32$ and evaluate the first 64 tokens of each document. We also evaluate only one passage from each document as a further cost reduction technique and to reduce potential covariance effects, as it seems likely that the memorization of two sequences from the same document are correlated.

2.2 Threat Model

Throughout this paper, we assume that an engineer is looking to train a large language model with billions of parameters on a dataset, and that there is a small subset of the dataset that would be undesirable to have the model memorize. The engineer, therefore, wishes to be able to accurately predict whether or not this subset of the training data will be memorized by the fully-trained model by expending a relatively small amount of compute. Following the literature on scaling laws [26, 32], we assume that the cost of training a model is approximately

$$C = 6 \times \text{[\# Params]} \times \text{[\# Tokens]} \quad (1)$$

and that the engineer has a computing budget that allows them to perform substantial testing before performing the full model training run.

2.3 Predicting Memorization

We can treat a smaller model's memorization of a sequence, or lack thereof, as a predictor for the memorization behavior of a larger

Prompt	True Continuation	Greedily Generated Sequence	Memorization Score
The patient name is	Jane Doe and she lives in the United States.	John Doe and he lives in the United Kingdom .	$\frac{0+1+1+0+1+1+1+1+0+1}{10} = 0.7$
Pi is defined as	the ratio of the raidus of a circle to its	a famous decimal that never enters a repeating pattern .	$\frac{0+0+0+0+0+0+0+0+0+0}{10} = 0$
The case defendant is	Billy Bob. They are on trial for tax fraud	Billy Bob . Are they really on trial for tax	$\frac{1+1+1+0+0+0+0+0+0+0}{10} = 0.3$
The case defendant is	Billy Bob. They are on trial for tax fraud	Billy Bob . They are on trial for tax fraud	$\frac{1+1+1+1+1+1+1+1+1+1}{10} = 1$

Table 1: Examples of memorization score calculation with different prompts. Note that these are provided for illustrative purposes and are not from the actual training data. The final example demonstrates a 4-elicitable string.

model. Whether the interested model did memorize the sequence is the ground truth label, and the smaller model's behavior is the prediction.

For example, if a smaller model memorized a sequence and the larger model did not, we can think of this case as a false positive. Likewise, if both models memorized the sequence, then the smaller model's prediction was a true positive. Models not memorizing the target sequence are negative cases.

This "prediction" by the smaller model compared against the ground truth allows us to calculate classification metrics such as precision and recall. In this case, *precision* tells us how many of the sequences memorized by the smaller model are also memorized by the larger model. *Recall* conveys the percentage of sequences memorized by the larger model that are also memorized by the smaller model. The same framing can also be applied when analyzing across time—where we compare the memorized sequences at a certain intermediate checkpoint, and wish to predict which sequences will be memorized by the completed model.

As the engineer's sole concern is to avoid memorization on an undesirable subset (see Section 2.2), false negatives and false positives in predicting memorization have very different impacts on their workflow: a false positive (i.e. incorrectly predicting that a model will memorize the undesirable subset) results in throwing away a cheap model that could have been fruitfully continued to train the final model, while a false negative (i.e. incorrectly predicting that a model will not memorize the undesirable subset) results in the costly training of a full model that could leak sensitive samples from the training dataset. We are therefore primarily interested in assessing the *recall* of the predictors and will tolerate a low precision if it comes with a high recall. We explore the tradeoffs in these costs in Section 3.

Primarily interested in Recall.

2.4 Choice of Models and Datasets

At the time of writing, the only publicly-available pretrained LLM scaling suites trained on fully public training data are EleutherAI's GPT-Neo [6, 7, 53] and Pythia models [5], and Cerebras systems' Cerebras-GPT [18]. All of these suites were trained on the Pile [4, 21]. Additionally, we were able to obtain access to the ROOTS dataset [33, 36] that the BigScience Workshop's BLOOM [46] model was trained on. Of these model suites, we choose to use Pythia because **(a):** All Pythia models saw data samples in the exact same order, **(b):** the training data differs slightly across the GPT-Neo models, **(c):** some BLOOM models only have three partially-trained checkpoints, and **(d):** Cerebras-GPT models don't provide partially-trained checkpoints.

The computational cost of many of the experiments we run is quite large. Consequently, we are unable to evaluate every partially-trained model checkpoint in the Pythia suite.[1] For most of our experiments, we choose to evaluate seven checkpoints spaced evenly throughout training. Specifically, we evaluate on checkpoints trained for $(23 \cdot 10^6)$, $(44 \cdot 10^6)$, $(65 \cdot 10^6)$, $(85 \cdot 10^6)$, $(105 \cdot 10^6)$, $(126 \cdot 10^6)$, and $(146 \cdot 10^6)$ sequences respectively, where these checkpoints approximately correspond to 7 checkpoints evenly spaced throughout training.

3 UNPREDICTABLE MEMORIZATION ACROSS SCALES

By far, the most common type of scaling law to study (and indeed, the origin of the term itself) is looking at how performance for very large models can be predicted based on performance of much smaller models. Fully-trained smaller model variants are independently useful as artifacts and can be applied in resource-constrained environments in place of larger models. Therefore, when projecting the characteristics of higher-compute model runs via scaling studies, training smaller model variants for this purpose is an actively desirable by-product, in contrast to the alternative of producing many shorter-training-duration checkpoints of the same single large architecture to extrapolate properties of a final full run. Therefore, the first question we seek to answer is: can an LLM's memorization behavior be predicted across model scales?

To evaluate how productive training small models can be for the purpose of predicting which datapoints will be memorized by large models, we subset our data to the sequences with a memorization score of 1 (meaning all 32 target tokens were produced accurately by the smaller model). Then, we look at the correlations between each pair of fully-trained model sizes for which sequences are memorized. The results are shown in Figure 1.

We see a sharp decline in correlation between which sequences are memorized by smaller models and the 12B model as the gap between the model sizes increases. Unfortunately, we find that these low correlation scores cause the set of sequences memorized by small models to have very poor predictive power in terms of what sequences will be memorized by a larger model. We also measure precision and recall of fully-memorized sequences using each smaller model to predict the memorization of the 12B model as shown in Table 2. Although the *precision* is high for all models (see Section 2.2), we are more interested in achieving a high recall than a high precision. The recall is incredibly low across the board, with even the 1.4B parameter model only achieving a recall of 0.554 when

[1] The cost of doing so would be comparable to the cost of training the models in the first place.

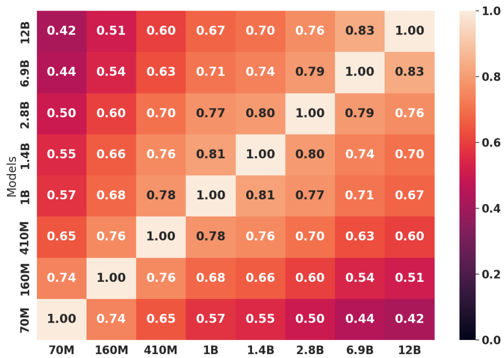


Figure 1: A heat map for visualizing the correlation between sequences memorized by different sizes. All models are fully trained.

trying to forecast the behavior of a model an order of magnitude larger.[2]

Model Size	Precision	Recall
70M	0.956	0.197
160M	0.948	0.289
410M	0.940	0.401
1.0B	0.931	0.512
1.4B	0.926	0.554
2.8B	0.909	0.658
6.9B	0.884	0.795
12B	—	—

Table 2: Precision and Recall when using each model to predict which sequences would be memorized by the 12B parameter model. For example, 95.6% of the sequences memorized by the 70M model were also memorized by the 12B model, but those only accounted for 19.7% of the sequences that the 12B model memorized.

Our findings suggest that using smaller model runs to forecast the memorization of larger models is not accurate. Due to the low recall, practitioners cannot use a small model's lack of memorization of a given sequence as a strong guarantee that their larger model will not memorize that same sequence. We therefore do not recommend using smaller model runs for this task, and seek to provide a setup that grants practitioners more assurances and a better compute tradeoff.

4 MEMORIZATION WITHIN TRAINING

The second question we seek to answer is: can an LLM's memorization behavior be predicted ahead of time within a training run? In Section 4.1, we provide evidence that such predictions are possible, and in Section 4.2 we investigate the effects of duplicated sequences within the training dataset on memorization behavior.

[2] Typical use-cases are to use smaller models to predict the behavior of models one to two orders of magnitude larger, see Chowdhery et al. [15], Rae et al. [43], Scao et al. [47].

4.1 Predictable Memorization

Despite the negative results of the previous section, we do find an unusual context in which practitioners training models can reliably test the memorization of a large language model before fully committing resources to training it. As Biderman et al. [5] found, location within the training data does not impact whether a particular sequence is memorized. Therefore, we propose that those concerned about the memorization of particular strings should move them early during training. Thus practitioners may have an early warning signal for detecting memorization of undesired sequences. In surprising contrast to the results in Section 3, we find that we can predict which points will be memorized by a fully-trained model by looking at a partially-trained model checkpoint very early during training.

In Figure 2, we show a correlation heatmap between which sequences are memorized by different checkpoints of the same model. We only look at memorization of the first 23 million sequences, as that is the data that our least-trained model checkpoint has seen.

Seq Num	Precision	Recall
$23 \cdot 10^6$	0.500	0.918
$44 \cdot 10^6$	0.575	0.915
$65 \cdot 10^6$	0.641	0.913
$85 \cdot 10^6$	0.711	0.911
$105 \cdot 10^6$	0.809	0.916
$126 \cdot 10^6$	0.916	0.943
$146 \cdot 10^6$	—	—

High Recall

Table 3: Precision and recall for predicting which sequences would be memorized by the fully-trained model from a partially-trained checkpoint. Current table specifies results for 12B. For both the 6.9B and the 12B model, we see that early checkpoints have high recall, ensuring that analysts can rely on predictions of non-memorization.

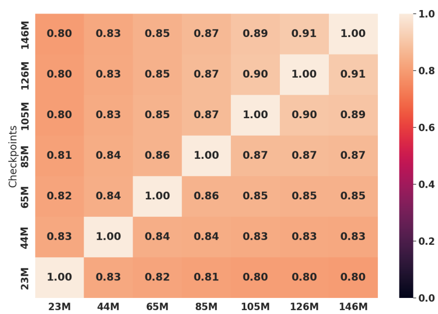
As shown in Table 3, recall consistently remains high over the course of training, even at the first checkpoint that we measure. This indicates that if a sequence would be memorized by the final model, there is a high likelihood that that sequence will be among those memorized by an intermediate checkpoint, thus providing a signal to an engineer seeking to predict what the final model memorizes ahead of time.

We also observe that precision rises over training, seemingly indicating that early model checkpoints generate more false positives, or memorize sequences that are later "forgotten" in the final model, than later checkpoints. However, Section 2.2 discusses why this outcome is more desirable than low recall.

4.2 Data Duplication and Memorized Sequences

Our consistently high recalls over the course of training present a curious picture of model learning dynamics, where models do not tend to forget memorized sequences. While this exact experimental set-up and context is novel, it is still in tension with existing literature [22, 29, 51]. Conventional wisdom would suggest that memorization is high immediately after encountering a sequence,

(a) 70M Parameter Model



(b) 1.4B Parameter Model

(c) 12B Parameter Model

Figure 2: Heat maps visualizing the correlation between which sequences are memorized by different checkpoints. Plots for other Pythia models can be found in Appendix A.

but decays over the course of training, with some memorized sequences being forgotten as the model encounters more data. To confirm our results, we examine the percentage of data points seen and memorized by the 23M sequence checkpoint that are memorized by subsequent checkpoints. We find that while there is an initial dip in the percentage of memorized sequences retained, it consistently climbs up over the course of the rest of training.

As a concrete example, Figure 3 shows that for the 12 billion parameter model, about 10% of the sequences memorized after 23

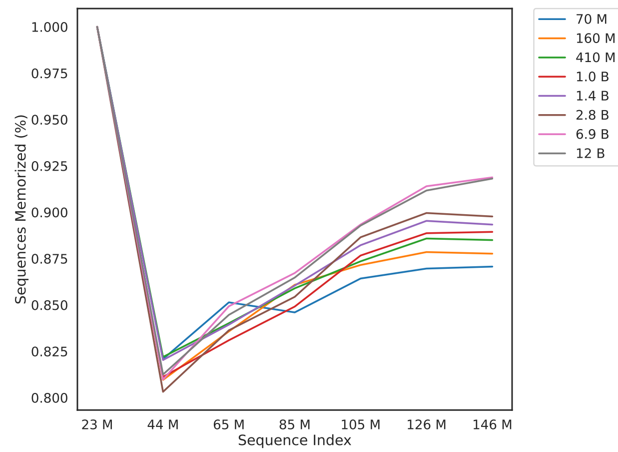


Figure 3: Percentage of sequences memorized by the 23×10^6 sequence checkpoint that remain memorized by subsequent checkpoints, for each model size.

million sequences are *no longer* memorized by the model after 44 million sequences, but are *again* memorized after 146 million sequences. We hypothesize that this is due to the fact that the Pile dataset is deliberately duplicated, while the aforementioned studies look at the memorization of data that occurs only once in the training corpus. This can be understood as a form of continual learning in large language models [48] via implicit experience replay [38].

While memorization is generally undesirable, in this context the "rememorization" phenomenon increases the reliability with which an engineer can predict what sequences will be memorized late in training. The potential cost of such a property is that sequences that were not memorized the first time through the data will be memorized in future iterations, decreasing the recall of the predictions. We term this "new memorization" and find that it is generally not the case, as shown in Table 4. The memorization rate conditional on not being memorized by the first checkpoint is minuscule by comparison to the base rate.

Model Size	Base Rate	New Rate
70M	0.0031	0.00008
160M	0.0047	0.00015
410M	0.0066	0.00024
1.0B	0.0085	0.00034
1.4B	0.0093	0.00038
2.8B	0.0114	0.00049
6.9B	0.0145	0.00068
12B	0.0126	0.00078

Table 4: End-of-training memorization rate for the dataset as a whole ("base"), and for data seen in the first 23 M sequences but not memorized during it ("new"). This indicates memorized sequences are much more likely to be memorized "right away" rather than much later during training.

5 SCALING LAWS

Having established the empirical results in the previous section, we now examine our results through the lens of computational efficiency and scaling laws, where the aim is to achieve the most reliable results for the least expense. To achieve this, we examine how well models of various sizes and number of training steps predict which sequences will be memorized **by the fully trained 12B parameter model**. This is in notable contrast to Section 4, where partially-trained models are only compared to fully-trained models of the same size. As a visual aid, models with the same size are colored the same.

5.1 Unusual Scaling

In the overwhelming majority of prior work on scaling laws [7, 8, 15, 32, 37, 41, 43, 47], including scaling studies targeting memorization [13, 25, 51], plots of quantities of interest vs compute are linear on a log or log-log plot. We find that this is not the case in our setup for both precision and recall (see Figure 8 in the Appendix for all plots).

The scaling data for precision is extremely anomalous. Not only are the plots non-linear, we find that the behavior of the 12B partially trained model is extremely out-of-line with the behavior of smaller models. This is best seen on the linear-linear plot (Figure 4).

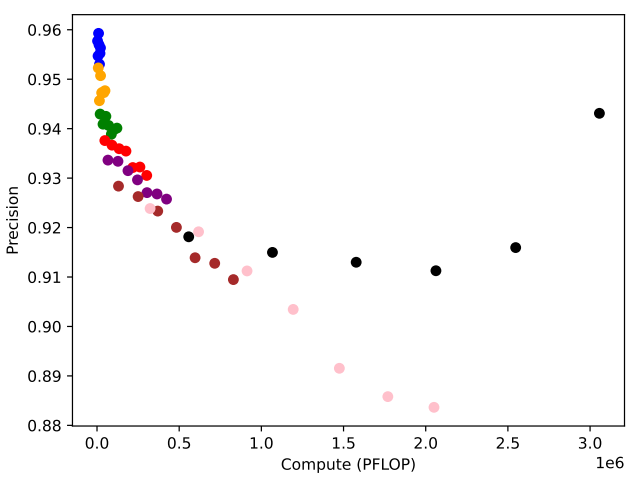


Figure 4: Scaling curve for precision on a linear-linear plot.

Despite the fact that there is a high-level pattern in the scaling laws curve for recall, a careful look at the data indicates unusual behavior. In the low-compute regimes, which are of most interest to engineers looking to minimize the cost of creating a prediction of the behavior of large models before they are trained, we see a consistent pattern of larger models being better than smaller models for a fixed compute budget. However, as the amount of compute expended scales, this is no longer the case. Starting at around 1% the budget of the fully trained model, equicompute models perform the same regardless of the number of parameters. Starting at around 10% the budget of the fully trained model, the *smallest* model trained for this compute budget becomes the best predictor of memorization in the fully trained model.

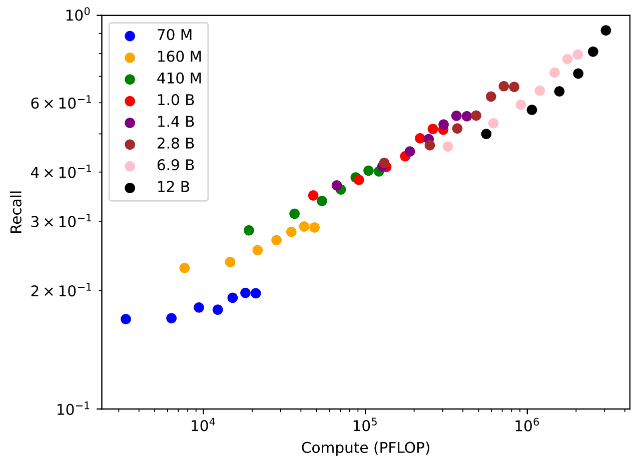


Figure 5: Scaling curve for recall on a log-log plot. For curves with linear-linear and log-linear axes, see Figure 8 in the appendix.

5.2 Emergent Memorization

We also see evidence of "emergent" or "semi-emergent" behavior as model scale increases. In the literature on emergent behavior in large language models [9, 20, 49, 54], the term refers to when a large model's performance on a task is substantially different from the extrapolated value of curves fit to the behavior of smaller models. Often, but not always, this occurs when performance goes from near-zero to meaningful. While our situation is not totally analogous, one can similarly consider "emergent memorization" to occur when data is memorized by large models which cannot be predicted based on the memorization behavior of smaller models. Since, by definition, emergent behavior implies that smaller-scale model behaviors are qualitatively different to those of larger models, this can pose challenges for traditional scaling laws or for extrapolating model behavior to models orders of magnitude larger. As a result, we suggest that this is an important area for further study, including expanding the scope of our work to models larger than 12B parameters. At scales larger than 12B, new things may "emerge".

5.3 Takeaways for Engineers

As discussed in Section 2.2, the primary point of interest to engineers is to predict the behavior of a large language model before it is trained. Such predictions should be grounded in low-cost regimes such as the behavior of trained "test" models that are at least an order of magnitude smaller than the target model. Consistent with the results in Sections 3 and 4, we find that when holding the compute budget fixed it is desirable to use as large a "test" model as possible when working with models substantially cheaper than the final model.

We further note that sequences seen by the model in late stages of training are more likely to be memorized (Figure 3). Consequently, we recommend that engineers looking to predict which sequences will be memorized by their model to front-load the training data with sequences whose memorization would be undesirable, and to train as large of a model as is feasible for a small number of steps.

6 ROBUSTNESS TO THRESHOLDING CHOICES

In Section 3 and Section 4.1, we subset the data to the sequences with a "memorization score" of 1 (i.e., sequences that are fully memorized under previous works' definition). This approach labels all sequences with more than 32 tokens memorized as equally memorized, despite the fact that in reality some will have a much longer accurately reproduced continuation than others. In this section we explore whether that effects our results.

First, we examine the shape of the distribution of memorization scores. We had originally assumed that the answer would be an (approximately) exponential distribution, under the assumption that LLMs had a constant "memorization rate" for correctly predicting each subsequent token. Our assumption was that this "memorization rate" would be based on model size, and that it was the primary determinant of overall memorization score distribution. This would be potentially problematic for our study, as the 32-token memorized sequences would dominate the set of memorized sequences.

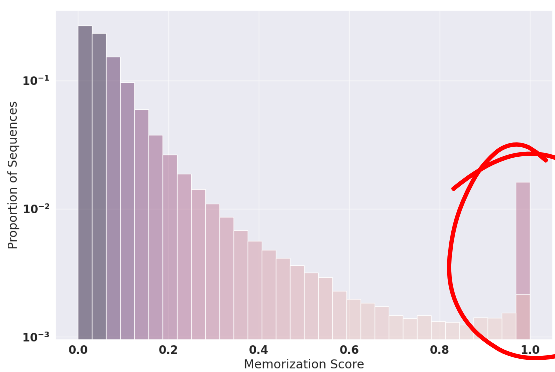


Figure 6: Distribution of memorization scores for 12B parameter model. For all upcoming sections in this paper, "memorized" is defined as $score = 1$.

However, upon examining the distribution of memorization scores for the largest Pythia models, it was immediately clear that this cannot be the case. As shown in Figure 6, there is a very evident spike in the memorization score distribution at $score = 1$. Exponential distributions are thin-tailed distributions, and while they would have a spike at $score = 1$, it is not possible for them to have such a large spike. The effect shown in Figure 6 can only occur in *thick-tailed* distributions, such as the power law distribution.

This is a good sign for our analysis, as it means that the typical memorized datapoint in fact has a much larger number of memorized tokens than the 32 token threshold we were worried about. We also replicate Table 2 with the doubled threshold and find roughly the same results.

We also find the same results on our scaling laws plots, as seen in the appendix.

Model Size	Precision	Recall
70M	0.949	0.140
160M	0.941	0.222
410M	0.931	0.334
1.0B	0.922	0.451
1.4B	0.918	0.497
2.8B	0.900	0.611
6.9B	0.872	0.775
12B	—	—

Table 5: Precision and Recall when using each model to predict which sequences would be memorized by the 12B parameter model. This table requires twice as many tokens to match to be considered memorized, but otherwise is a replication of Table 2.

Even Considering twice num tokens as memorization also gives similar trends.

7 LIMITATIONS AND FUTURE WORK

Our work constitutes the first steps towards developing a way to predict what data will be memorized by a large language model before that model is trained, but has several limitations and opens opportunities for exciting future work. The most important of these are:

Are we measuring the correct thing? The definition of memorization we use is derived from what is currently popular in the academic literature, but it is unclear if it is the best definition to use. We believe k-elicitation to be well-grounded in privacy concerns of language models, but other metrics such as memorization score may be more natural when studying the *dynamics* of memorization in training.

Does this generalize to other models? We report our experiments on the Pythia suite, because it is the only current language modeling suite suitable for such work. However this leaves open many questions about whether our results generalize to models trained with different hyperparameters or different data. We intend to run our analyses on the deduplicated Pythia suite, but this gap points to the need for more reproducible, public dataset model releases to advance research on memorization.

What about the data contents? Our work does not take the actual content of the training data into account at any point in time: we are looking exclusively at predicting memorization based on whether other cheaper models memorize the content. Future work looking into whether there are properties of the training text that predict memorization of that text could be quite illuminating.

8 CONCLUSION

We propose a novel setting for forecasting model memorization prior to train-time, while minimizing the compute required to make this forecast. We present analyses on the two most natural setups for extrapolation: using **fully-trained small models** and **partially-trained checkpoints of the final model** to compare and predict memorization of the final large model. We find that using smaller models for this task is not viable, and that partial checkpoints of an existing model are much more effective predictors of final memorization behavior. We additionally present a novel formulation of "memorization score" not commonly used in existing literature, and utilize this score to investigate the distribution of "partially-memorized" data points to derive further insights and intuitions on the dynamics of memorization. We hope that our focus on extrapolation will be compelling for future work to continue studying, and that our analyses inform deep learning practitioners on methods to understand and reduce memorization while training large language models.

REFERENCES

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[2] Ahdritz, G., Bouatta, N., Kadyan, S., Xia, Q., Gerecke, W., O'Donnell, T. J., Berenberg, D., Fisk, I., Zanichelli, N., Zhang, B., et al. Openfold: Retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *bioRxiv*, 2022.

[3] Anil, R., Ghazi, B., Gupta, V., Kumar, R., and Manurangsi, P. Large-scale differentially private bert. *arXiv preprint arXiv:2108.01624*, 2021.

[4] Biderman, S., Bicheno, K., and Gao, L. Datasheet for the pile. *arXiv preprint arXiv:2201.07311*, 2022.

[5] Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*, 2023.

[6] Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: large scale autoregressive language modeling with mesh-tensorflow. *GitHub*, 2021. URL https://www.github.com/eleutherai/gpt-neo.

[7] Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, 2022.

[8] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. URL https://arxiv.org/abs/2005.14165.

[9] Caballero, E., Gupta, K., Rish, I., and Krueger, D. Broken neural scaling laws. *arXiv preprint arXiv:2210.14891*, 2022.

[10] Cao, Y., Chen, Z., Belkin, M., and Gu, Q. Benign overfitting in two-layer convolutional neural networks. *Advances in neural information processing systems*, 35: 25237–25250, 2022.

[11] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 267–284, 2019.

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

[13] Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

[14] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[15] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[16] Crowson, K., Biderman, S., Kornis, D., Stander, D., Hallahan, E., Castricato, L., and Raff, E. Vqgan-clip: Open domain image generation and editing with natural language guidance. *arXiv preprint arXiv:2204.08583*, 2022.

[17] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[18] Dey, N., Gosal, G., Zhiming, Chen, Khachane, H., Marshall, W., Pathria, R., Tom, M., and Hestness, J. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster, 2023.

[19] Fried, D., Aghajanyan, A., Lin, J., Wang, S., Wallace, E., Shi, F., Zhong, R., Yih, W.-t., Zettlemoyer, L., and Lewis, M. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*, 2022.

[20] Ganguli, D., Hernandez, D., Lovitt, L., Askell, A., Bai, Y., Chen, A., Conerly, T., Dassarma, N., Drain, D., Elhage, N., et al. Predictability and surprise in large generative models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1747–1764, 2022.

[21] Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The Pile: an 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. URL https://arxiv.org/abs/2101.00027.

[22] Graves, L., Nagisetty, V., and Ganesh, V. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.

[23] Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D. M., Schulman, J., Amodei, D., and McCandlish, S. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. URL https://arxiv.org/abs/2010.14701.

[24] Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021. URL https://arxiv.org/abs/2102.01293.

[25] Hernandez, D., Brown, T., Conerly, T., DasSarma, N., Drain, D., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Henighan, T., Hume, T., et al. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*, 2022.

[26] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[27] Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P. S., and Zhang, X. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[28] Ippolito, D., Tramèr, F., Nasr, M., Zhang, C., Jagielski, M., Lee, K., Choquette-Choo, C. A., and Carlini, N. Preventing verbatim memorization in language models gives a false sense of privacy. *arXiv preprint arXiv:2210.17546*, 2022.

[29] Jagielski, M., Thakkar, O., Tramer, F., Ippolito, D., Lee, K., Carlini, N., Wallace, E., Song, S., Thakurta, A., Papernot, N., et al. Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*, 2022.

[30] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[31] Kandpal, N., Wallace, E., and Raffel, C. Deduplicating training data mitigates privacy risks in language models. *arXiv preprint arXiv:2202.06539*, 2022.

[32] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

[33] Laurençon, H., Saulnier, L., Wang, T., Akiki, C., del Moral, A. V., Le Scao, T., Von Werra, L., Mou, C., Ponferrada, E. G., Nguyen, H., et al. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[34] Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[35] McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[36] McMillan-Major, A., Alyafeai, Z., Biderman, S., Chen, K., De Toni, F., Dupont, G., Elsahar, H., Emezue, C., Aji, A. F., Ilić, S., et al. Documenting geographically and contextually diverse data sources: The bigscience catalogue of language data and resources. *arXiv preprint arXiv:2201.10066*, 2022.

[37] Mikami, H., Fukumizu, K., Murai, S., Suzuki, S., Kikuchi, Y., Suzuki, T., Maeda, S.-i., and Hayashi, K. A scaling law for synthetic-to-real transfer: How much is your pre-training effective? *arXiv preprint arXiv:2108.11018*, 2021. URL https://arxiv.org/abs/2108.11018.

[38] Ostapenko, O., Lesort, T., Rodríguez, P., Arefin, M. R., Douillard, A., Rish, I., and Charlin, L. Continual learning with foundation models: An empirical study of latent replay. In *Conference on Lifelong Learning Agents*, pp. 60–91. PMLR, 2022.

[39] Popov, V., Kudinov, M., Piontkovskaya, I., Vytovtov, P., and Nevidomsky, A. Differentially private distributed learning for language modeling tasks. *arXiv preprint arXiv:1712.07473*, 2017.

[40] Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[41] Pu, J., Yang, Y., Li, R., Elibol, O., and Droppo, J. Scaling effect of self-supervised speech models. *Proc. Interspeech 2021*, pp. 1084–1088, 2021.

[42] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

[43] Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[44] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[45] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

[46] Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[47] Scao, T. L., Wang, T., Hesslow, D., Saulnier, L., Bekman, S., Bari, M. S., Bideman, S., Elsahar, H., Muennighoff, N., Phang, J., et al. What language model to train if you have one million gpu hours? *arXiv preprint arXiv:2210.15424*, 2022.

[48] Scialom, T., Chakrabarty, T., and Muresan, S. Continual-t0: Progressively instructing 50+ tasks to language models without forgetting. *arXiv preprint arXiv:2205.12393*, 2022.

[49] Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[50] Tirumala, K., Markosyan, A., Zettlemoyer, L., and Aghajanyan, A. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.

[51] Tirumala, K. N. B., Markosyan, A. H., Zettlemoyer, L., and Aghajanyan, A. Memorization without overfitting: Analyzing the training dynamics of large language models. *ArXiv*, abs/2205.10770, 2022.

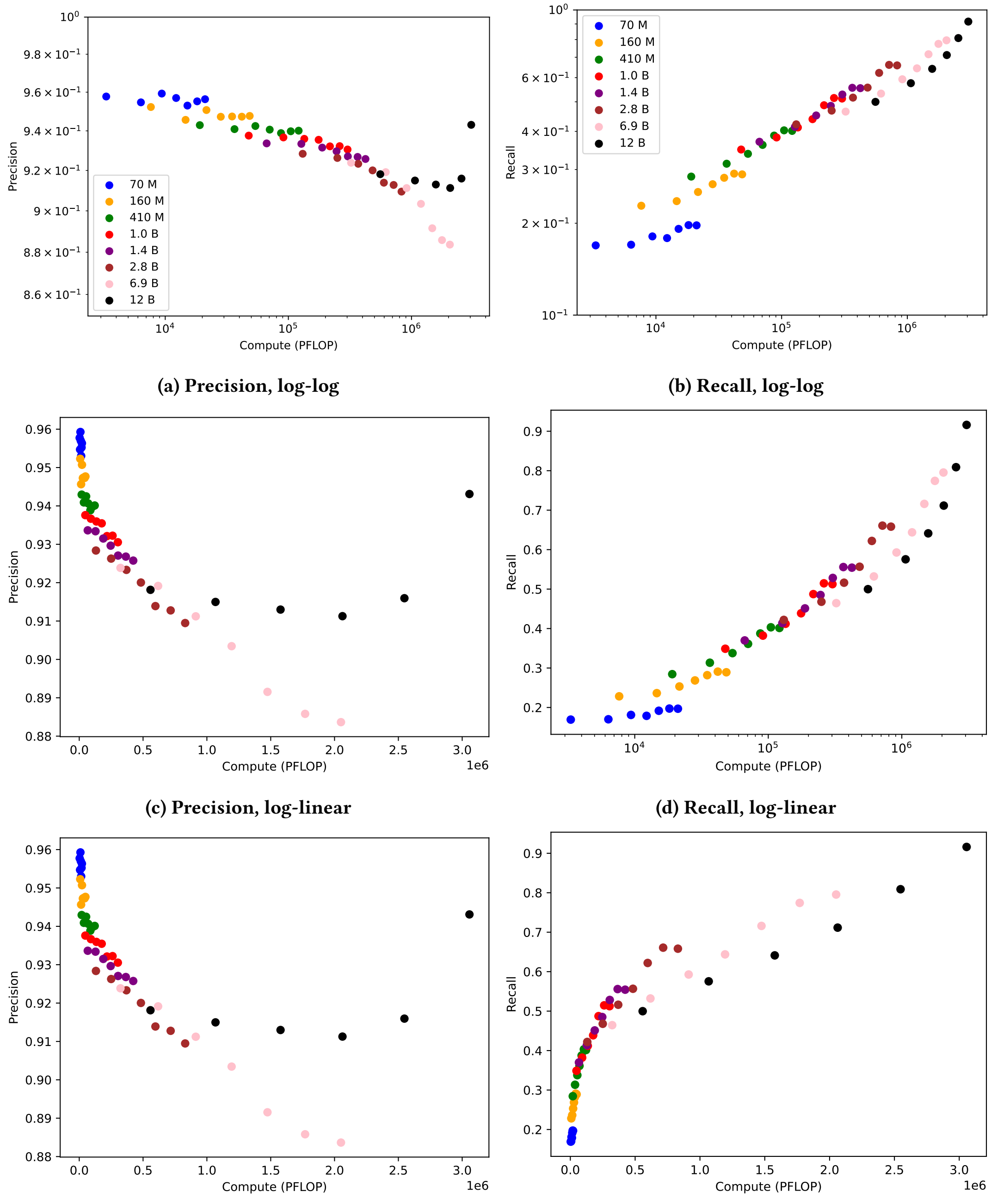
[52] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[53] Wang, B. and Komatsuzaki, A. GPT-J-6B: a 6 billion parameter autoregressive language model, 2021.

[54] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[55] Xu, F. F., Alon, U., Neubig, G., and Hellendoorn, V. J. A systematic evaluation of large language models of code. *arXiv preprint arXiv:2202.13169*, 2022.

A ADDITIONAL FIGURES

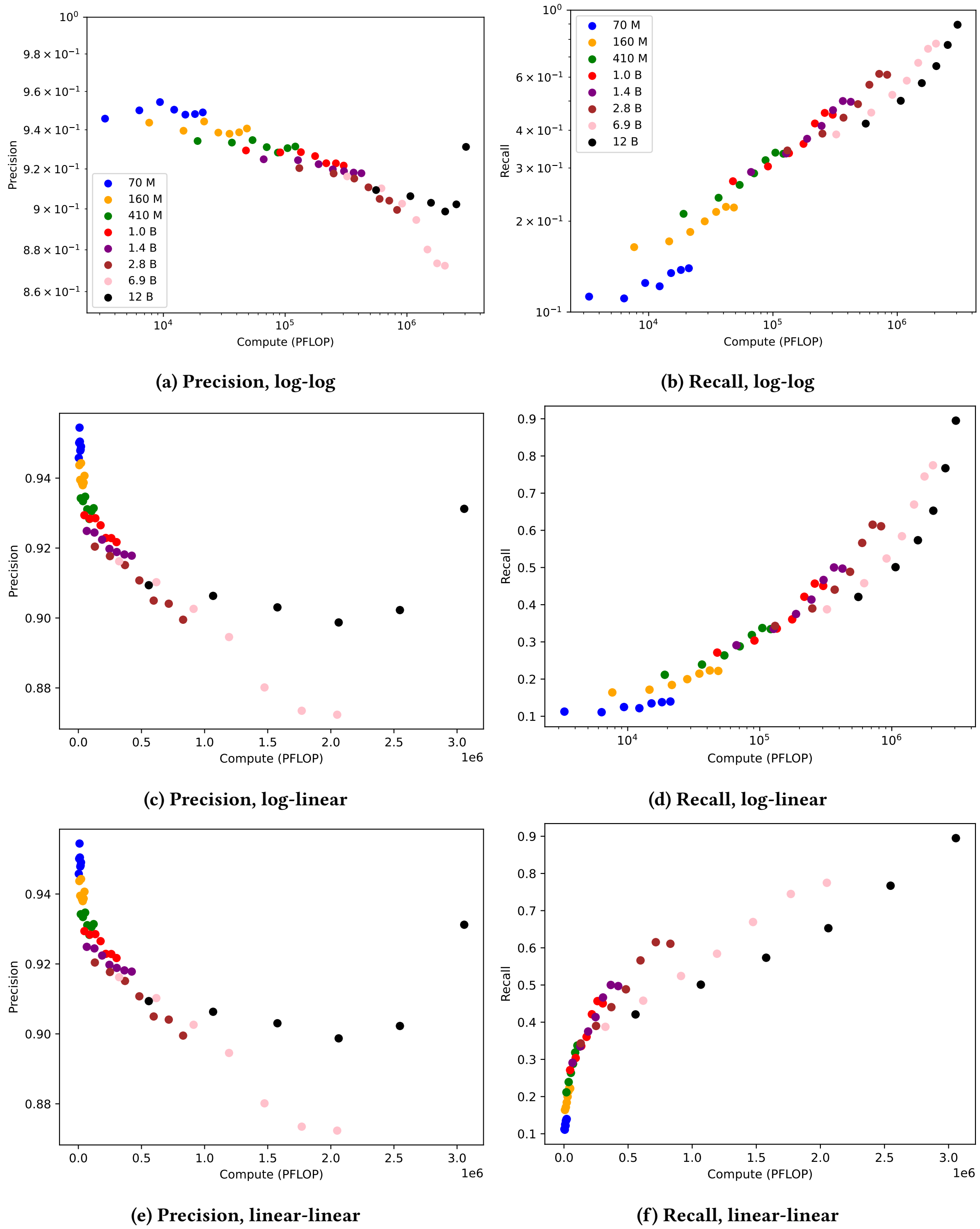


(a) Precision, log-log **(b) Recall, log-log**

(c) Precision, log-linear **(d) Recall, log-linear**

(e) Precision, linear-linear **(f) Recall, linear-linear**

Figure 7: Scaling laws curves.



(a) Precision, log-log

(b) Recall, log-log

(c) Precision, log-linear

(d) Recall, log-linear

(e) Precision, linear-linear

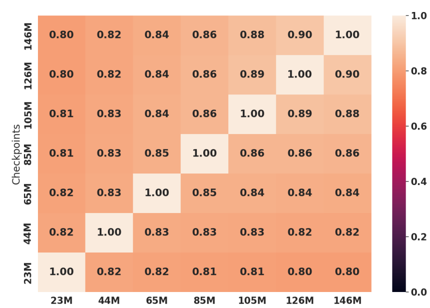
(f) Recall, linear-linear

Figure 8: Scaling laws curves with twice as many tokens required to be correctly reproduced to be considered "memorized."

(a) 70M Parameter Model

(b) 160M Parameter Model

(c) 410M Parameter Model

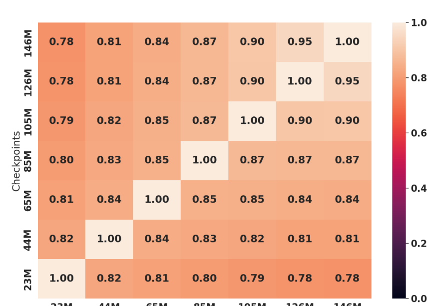


(d) 1.0B Parameter Model

(e) 1.4B Parameter Model

(f) 2.8B Parameter Model

(g) 6.9B Parameter Model



(h) 12B Parameter Model

Figure 9: Heat maps visualizing the correlation between which sequences are memorized by different checkpoints.