How Much Information Can We Fit in a Transformer?

Anonymous authors
Paper under double-blind review

Abstract

How efficiently can transformers store factual knowledge? The answer to this question has implications for both model design and understanding the limits of transformer-based systems. Prior studies observe a linear relationship between the number of facts memorized and the number of model parameters required; substantially worse than what theoretical analyses predict is achievable. We re-examine empirical scaling: through rigorous and systematic experiments, including extensive hyperparameter search via Bayesian optimization, we determine the smallest model that can perfectly memorize a given set of facts, and measure how this minimum size scales with the number of facts. On synthetic datasets of random, independent (subject, relation, object) triples—e.g., `(Paris, capital_of, France)`—drawn at random with no exploitable structure, we find that memorization capacity, the size of the largest dataset a model memorizes completely, grows exponentially with the transformer dimension: the minimum transformer dimension $d_{\min}$ required to memorize N facts grows logarithmically in N, and across 11 dataset sizes spanning $N = 45$ to $49{,}920$ a lack-of-fit F-test does not reject the logarithmic fit (with an offset) but it does reject all other tested alternatives including power-law. Along the depth axis, however, at fixed width, the maximum memorizable dataset size grows sublinearly with the number of layers over the measured range. We further extend our exploration to structured data containing symmetric relations or two-hop composition. In this setting, we find that transformers exploit these regularities to further improve capacity. All code is available to reproduce the results at `[redacted]`.

1 Introduction

Large language models (LLMs) exhibit a remarkable ability to memorize facts and associations, sometimes to the point of verbatim recall (Carlini et al., 2021; 2023; Petroni et al., 2019). As transformer models scale to trillions of parameters, critical questions arise: What is the maximum number of associations a model of given size can memorize? How does this capacity scale with model size? Answering these questions enables practitioners to select appropriate architectures for knowledge-intensive tasks and predict when transformers will successfully memorize versus fail to learn training data due to insufficient capacity (Li et al., 2025). Our central research questions are: (1) how does the minimum transformer model dimension required to perfectly memorize N facts scale with N? and (2) at fixed model dimension, how does the maximum number of memorizable facts scale with the number of layers L?

Prior empirical studies of transformer memorization measure how many bits of factual knowledge are learned per parameter. Allen-Zhu & Li (2025) train GPT-2, LLaMA, and Mistral models on synthetic biographies composed of (name, attribute, value) tuples and establish, through an information-theoretic bit-complexity analysis, that sufficiently trained transformers store $\sim$2 bits of information per parameter. Morris et al. (2026) train GPT-style models on synthetic random sequences and report $\sim$3.6 bits per parameter. Lu et al. (2024), training Qwen-style LLMs directly on relational triples, do report approximately linear scaling between fact capacity and number of parameters, estimating that memorizing Wikidata would require $\sim$1000 billion parameters; however, their measurements use 50–200 training epochs while their own analysis indicates that capacity saturates only at around 1000 epochs and hyperparameters are only partially tuned at different model sizes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Theoretical work has established bounds on transformer memorization capacity. The result most directly relevant to our work is Kajitsuka & Sato (2025), who study the next-token prediction setting (a single label per input sequence, equivalent to ours) and prove that $\tilde{O}(\sqrt{N})$ parameters are sufficient for a transformer to memorize N facts, and that $\Omega(\sqrt{N})$ parameters are necessary for memorizing every $\{0, 1\}$-labeling of N inputs; these upper and lower bounds are the same up to log factors and are much better than the linear scaling observed in empirical studies. Other upper-bound constructions for related settings give larger parameter counts (Kajitsuka & Sato, 2024; Madden et al., 2025; Mahdavi et al., 2024; Kim et al., 2023) while Hopfield-network and Johnson–Lindenstrauss theoretical results predict transformer dimension needs to scale logarithmically with N for memorization in idealized settings; we discuss both in Section 2. Our results show that the standard fixed-depth softmax transformer can memorize N facts with the transformer dimension scaling as $\ln N$.

We believe that the gap with prior empirical work has one main source. Prior studies that report a linear relationship between parameter count and the number of memorized facts (Lu et al., 2024; Morris et al., 2026) use limited hyperparameter search and limited training compute, both of which prevent the model from reaching its maximum information capacity at larger dataset sizes (Section 2). Allen-Zhu & Li (2025) demonstrate that the number of times each fact is seen during training (i.e., *exposure*) directly governs measured capacity, confirming that insufficient training duration causes underestimation of the model's information capacity.

In this work we show that standard transformers empirically achieve substantially better scaling than the linear relationship between number of parameters and number of facts reported in prior empirical work (Allen-Zhu & Li, 2025; Lu et al., 2024). We train transformers on synthetic relational triples (Bordes et al., 2013) with uniformly random structure, since we aim to test the model's memorization capability while controlling for generalization. We find that the number of facts N that a transformer can memorize[1] grows *exponentially* with transformer dimension. Equivalently, the minimum transformer dimension $d_{\min}$ required to memorize N facts grows logarithmically in N. On `Uniform` data, the logarithmic form is the only form that is not rejected (Section 4.1). At fixed width $d = 8$ and a fixed training budget, the maximum memorizable dataset size grows with depth, but sublinearly; a lack-of-fit F-test rejects proportional growth ($p = 7.8 \times 10^{-6}$), a straight line with an intercept ($p = 0.0025$), and superlinear forms ($p \leq 1.2 \times 10^{-5}$), while a logarithm and a power-law with exponent 0.65 are not rejected, and our data do not distinguish between them (Section 4.2).

In addition, we extend our analysis beyond unstructured uniformly-distributed data. For data with logical structure, such as data containing symmetric relations (if (s, r, o) then (o, r, s)) and two-hop compositional relations (if (a, r_1, b) and (b, r_2, c) then (a, r_3, c)), we find that transformers exploit these regularities to achieve improved scaling. To probe whether *distribution skew* alone affects capacity—a relevant question because real-world fact distributions are typically Zipf-like rather than uniform—we test power-law-distributed data, finding that they show similar scaling to uniformly-distributed data.

Our key contributions are:

- We demonstrate that memorization capacity grows exponentially with the transformer dimension: at fixed depth $L = 1$, the minimum transformer dimension d needed to memorize N facts grows logarithmically in N, a lack-of-fit F-test rejects the power-law and other alternatives.
- At fixed width and a fixed training budget, capacity grows sublinearly with the number of layers: linear and superlinear growth are rejected, and the sublinear forms we tested are not distinguished.
- We show that the gap with prior empirical work stems from training methodology: both insufficient training duration and limited hyperparameter optimization lead to suboptimal scaling trends.
- We characterize how relational structure and the shape of the data distribution affect memorization. Structured data (symmetric relations and two-hop compositions, defined in Section 3.6) leads to improved scaling by exploiting regularities; Moreover, power-law-distributed data shows a similar scaling behavior.

[1] I.e., the model achieves 100% training accuracy in predicting the object given the subject and relation.

These findings suggest that large language models may have substantially greater knowledge capacity than previous empirical estimates indicate, with implications for understanding how transformers store knowledge (Geva et al., 2021) and for predicting capacity requirements at scale.

2 Related Work

Scaling Laws. Empirical scaling laws characterize how language model performance varies with computational resources. Kaplan et al. (2020) established power-law relationships between loss and model parameters, dataset size, and training compute; Hoffmann et al. (2022) revised their compute-allocation recommendations, showing parameters and training tokens should scale equally. Henighan et al. (2020) extended scaling laws to multimodal domains. Extensive work on scaling laws has since followed, and we refer the reader to a survey (Li et al., 2025). These scaling laws characterize average performance (cross-entropy loss) rather than memorization capacity.

Empirical Memorization Studies. Earlier work by Tirumala et al. (2022) showed that larger transformer models memorize training data faster and can memorize a larger fraction of the training data before overfitting, i.e., before their validation loss begins to increase. Three recent studies measure transformer memorization capacity, with complementary methodologies and findings. Lu et al. (2024) train Qwen-style LLMs on relational triples and report approximately linear scaling between parameters and fact capacity, estimating that memorizing Wikidata would require $\sim$1000 billion parameters; their scaling measurements use 50–200 training epochs, while their own analysis indicates that the fraction of facts recalled correctly continues to improve up to 1000 epochs, suggesting that longer training might yield better scaling behavior. Allen-Zhu & Li (2025) introduce an information-theoretic framework, training GPT-2, LLaMA, and Mistral models on synthetic biography datasets composed of (name, attribute, value) tuples—closely analogous to the relational triples studied by Lu et al. (2024) and in our work. They establish $\sim$2 bits/param for sufficiently trained transformers. They further show that the number of times each fact is seen during training critically affects measured capacity: with 1000 exposures per fact, transformers achieve 2 bits/param, but with only 100 exposures they achieve 1 bit/param. Morris et al. (2026) estimate capacity for GPT-style models at $\sim$3.6 bits per parameter. Crucially, the bits/param measurements of Allen-Zhu & Li (2025) and Morris et al. (2026) are measurements for sufficiently trained transformers for a specific dataset size, not assertions about how capacity scales with dataset size more broadly. Our work is complementary: rather than reporting a single bits-per-parameter constant, we fit a precise functional form $d_{\min}(N)$, the minimum transformer dimension that achieves perfect memorization, across a wide range of values of N, finding that $d_{\min}$ grows logarithmically in N across this range.

Theoretical Bounds on Memorization Capacity. Theoretical analyses of transformer memorization build on contextual mapping constructions (Yun et al., 2020). These results are typically worst-case bounds over arbitrary labelings, and assumptions vary across settings (e.g., the prediction task, label space, and separation conditions on inputs). Most directly relevant to our setting, Kajitsuka & Sato (2025) prove that $\tilde{O}(\sqrt{N})$ parameters are sufficient to memorize N sequences with a single label each in the next-token prediction setting, and that this bound is tight up to log factors. Earlier upper-bound constructions for related but distinct settings give different parameter-count scalings: $O(dnN + d^2)$ parameters for sequence-to-sequence with one-layer transformers (Kajitsuka & Sato, 2024), $\tilde{O}(d + n + \sqrt{nN})$ parameters for sequence-to-sequence with $O(n)$ layers (Kim et al., 2023), $\Theta(\zeta n)$ parameters for next-token *distribution* prediction (Madden et al., 2025), and $\Omega(Hn)$ parameters for multi-head attention memorizing arbitrary linearly-independent inputs (Mahdavi et al., 2024).

Nichani et al. (2025) study a synthetic factual-recall task with *fixed random embeddings* and prove that a single-layer transformer reaches perfect accuracy when either its self-attention or its MLP parameters scale linearly (up to log factors) with the number of facts N. Our embeddings are jointly trained, and the number of non-embedding parameters grow as $(\ln N)^2$ in N, rather than the linear $P \sim N$.

Our results on memorization capacity scaling with respect to transformer dimension are consistent with concurrent work by Ravfogel et al. (2026), who fit a logarithmic curve to the smallest embedding dimension that reaches 95% training accuracy on random bijections. However, they do not report a statistical test and do not compare the logarithmic fit with other forms such as a power-law. Our work,

on the other hand, systematically studies the scaling of memorization capacity across 11 dataset sizes and compares logarithmic, power-law and linear forms with a lack-of-fit F-test (Section 4.1).

Depth and memorization. Nichani et al. (2025) further prove that a single self-attention layer followed by an MLP stores facts at a rate proportional to its parameter count, and Chen & Zou (2024) find on synthetic sequence tasks that one attention layer suffices for memorization while reasoning and generalization need two or three layers. Neither result predicts what gradient training can achieve when layers are added at a fixed width and a fixed training budget. We measure this in our depth experiment.

Two theoretical results predict that storing N items in a d-dimensional space is achievable when $d \sim \log N$. The Johnson–Lindenstrauss lemma (Johnson & Lindenstrauss, 1984) establishes that $\Theta(e^{cd})$ near-orthogonal unit vectors fit in d-dimensional Euclidean space. Equivalently, $d = O(\log N)$ suffices to fit N such vectors. Modern Hopfield-network analyses (Ramsauer et al., 2021; Lucibello & Mézard, 2024) show that softmax attention, viewed as an associative memory over stored patterns, has retrieval capacity exponential in the pattern dimension. Both results concern idealized settings—vector geometry and a single softmax retrieval step over fixed patterns, respectively—rather than trained transformers with FFN, residuals, and learned embeddings.

3 Background and Methods

Throughout, we use N to denote the number of facts in the dataset, d for the transformer dimension (which equals the embedding dimension in our setting, and by standard convention), L for the number of layers, and P for the transformer (non-embedding) parameter count. The total number of parameters is $P + |V| \cdot d$, where $|V| \cdot d$ is the size of the embedding table; in our experimental setup $|V|$ scales with N.

Our goal is to investigate how transformer memorization capacity scales with model dimension (width) and number of layers (depth) under realistic training constraints.

We measure capacity along two axes: given a dataset, finding the smallest dimension d^* that achieves perfect memorization (100% accuracy); given a model, finding the largest dataset size N^* it can perfectly memorize. We optimize training hyperparameters in each setting to maximize measured capacity. For detailed mathematical formulation, see Appendix A.

We make a simplifying assumption that all facts have the form (s, r, o), where s is the subject entity, r is the relation, and o is the object entity. For example, the fact "Paris is the capital of France" can be expressed as $(\texttt{Paris}, \texttt{capital_of}, \texttt{France})$. This (s, r, o) representation is widely adopted in empirical and theoretical studies of memorization and factual recall in language models (Bordes et al., 2013; Petroni et al., 2019; Allen-Zhu & Li, 2025; Lu et al., 2024; Nichani et al., 2025). In this controlled setting, the model can only achieve high accuracy by memorizing facts. Mechanistic interpretability studies imply that transformers internally represent factual knowledge as structured associations: feed-forward layers function as key-value memories (Geva et al., 2021), specific neurons encode individual facts (Dai et al., 2022; Meng et al., 2022). Transformers decode facts using relation-specific linear transformations (Hernandez et al., 2024) and acquire this structured knowledge during pretraining by extracting and compressing entities and relations from text (Allen-Zhu & Li, 2024; Chang et al., 2024). The (s, r, o) abstraction corresponds to these representations. Further, it is easy to generate an arbitrary number of (s, r, o) triples with a high degree of control over their distribution and structure. Studying the ability of transformers to memorize triples facilitates our measurement of the fundamental capacity of the associative memory mechanisms that transformers employ for factual knowledge, enabling us to probe the model's memorization capacity independently of its information extraction ability.

3.1 Task Formulation

We measure memorization capacity using knowledge graph triples (s, r, o) where the subject s and relation r together uniquely specify the object o. Each input query is tokenized as a sequence of two input tokens (s, r) and the model must learn to map each input to its unique corresponding output token o. This task requires perfect memorization of all N associations in the dataset.

Definition (memorization and capacity). A trained model *memorizes* a fact (s, r, o) if, given the two-token input (s, r), its highest-probability output token is o (i.e., greedy decoding). A model *memorizes a dataset* of N facts if it memorizes every fact—that is, it reaches 100% training accuracy. *Capacity* is defined as the largest dataset size that a given model can memorize completely. The transformer width and depth scaling experiments measure this same quantity in two different ways. In the width scaling experiments, we fix the dataset size N and search for the smallest transformer model dimension d such that $\mathrm{Train}(d, 1, N; \theta)$ memorizes the dataset for some setting of hyperparameters θ (Eq. 1); this finds the smallest model whose capacity is at least N. Conversely, in the depth scaling experiments, we fix the model and search for the largest N such that $\mathrm{Train}(d, L, N; \theta)$ memorizes the dataset (Eq. 2); this measures the model's capacity directly. All reported values use the optimal hyperparameters found by the search described in Appendix B.

3.2 Model Architecture

We use the GPT-2 Radford et al. (2019) architecture: L transformer layers of self-attention with h heads, each followed by a feed-forward layer with hidden dimension $4d$. Entities and relations share a single learned embedding table of size $|V| \times d$, where $|V|$ grows with the number of entities n. We do not use positional embeddings; since all sequences have the fixed form (s, r, o), position is implicit in the input structure (preliminary experiments on this choice are described in Appendix E).

Following standard convention (Kaplan et al., 2020), the embedding dimension equals d, so total parameters split into embedding ($|V| \cdot d$) and non-embedding ($\Theta(Ld^2)$, attention and feed-forward weights); the latter is $P = 12d^2 + 15d$ for our single-layer, single-head architecture. We report P and total parameters separately throughout, since the embedding table is determined by vocabulary and P is determined by the architecture alone.

3.3 Hyperparameter Optimization

We optimize training hyperparameters using Bayesian optimization (BO) with tree-structured Parzen estimator (TPE) sampling via the `Optuna` library (Bergstra et al., 2011; Akiba et al., 2019). We stratify the search by learning-rate schedule (`linear` or `cosine`): a separate Bayesian optimization is run for each schedule and the best across schedules is reported. The search space is given in Table 2 (Appendix B); each candidate is evaluated on three seeds, with 60 completed trials per schedule for every study.

3.4 Training Protocol

We train in bfloat16 mixed precision using the Adam optimizer (Kingma & Ba, 2015) to minimize cross-entropy loss with L_2 weight decay obtained via Bayesian optimization. Training is full-batch: every fact appears in every iteration. We cap training at 40,000 iterations (excluding warmup)—up to 40,000 exposures per fact—based on our convergence analysis (Appendix E), with early stopping only after 100% accuracy holds for 50 consecutive iterations, i.e., after memorization is complete. That is 40$\times$ the 1,000-exposure saturation regime of Allen-Zhu & Li (2025).

3.5 Capacity Search Procedures

For each hyperparameter configuration we locate the boundary—d^* for width experiments at $L = 1$, or N^* for depth experiments at fixed width $d = 8$—using a two-phase search: geometric halving (width, starting from $d = 32$) or, for depth, doubling or halving the dataset size from an initial estimate until the first failure (or success if the starting estimate fails), then binary search to pinpoint it. The full procedure, including the upward-doubling fallback when $d = 32$ already fails, is in Appendix B.

3.6 Dataset Construction

For all experiments, we use synthetic datasets comprising (s, r, o) triples. We generate both unstructured and structured datasets to investigate whether transformers can exploit structure to achieve higher memorization capacity. To create a dataset, we specify two parameters: the number of entities n and the number of relations m.

We generate four datasets: `Uniform`, `Symmetric`, `2Hop`, `PowerLaw`.

To remain consistent with our deterministic full-batch methodology each batch contains every fact for all four datasets. For `Uniform`, `Symmetric`, and `2Hop`, each fact appears once per batch. For `PowerLaw` (Zipf with exponent $\alpha = 1.0$), each fact i appears $\max(1, \lceil k\, p_i \rceil)$ times per batch, where p_i is its Zipf probability and the multiplier $k = 1/p_N$ is chosen so that the rarest fact appears exactly once per batch. Frequent facts therefore appear more times per batch than rare ones; full implementation details are in Appendix D. We conduct experiments with $n \in \{15, 30, 60, 120, 260, 520, 1040, 2080, 4160, 8320, 16640\}$ entities and $m = 3$ relations across all dataset families (`Symmetric` up to $n = 4160$ and `2Hop` up to $n = 8320$). For `PowerLaw`, we only consider $n \in \{15, 30, 60, 120, 260, 520, 1040, 2080, 4160\}$ since k grows with n and the batch becomes prohibitively large otherwise.

Uniform Dataset (unstructured): For `Uniform`, we create a dataset such that, for any subject $s \in \{1, 2, \cdots, n\}$ and for any relation $r \in \{1, 2, \cdots, m\}$, there is a triple in the dataset (s, r, o) where the object o is uniformly sampled from the set $\{1, 2, \cdots, n\}$. For any subject-relation pair (s, r), we have a unique object o. Therefore, the dataset size is mn.

Symmetric Dataset (structured): For `Symmetric`, we process the subject-relation pairs (s, r) in the same order as `Uniform`. If no triple (o', r, s) is already in the dataset, we sample the object o uniformly from the entities not yet used with relation r, and add both (s, r, o) and its mirror (o, r, s) to the dataset (a single fact if $o = s$); otherwise, the fact for (s, r) was already added as a mirror. Every relation is therefore symmetric (e.g., if a is a sibling of b, then b is a sibling of a): if a triple (s, r, o) is in the dataset, then its counterpart (o, r, s) is also in the dataset.

2Hop Dataset (structured): For `2Hop`, we use $m = 3$ relations and dedicate the last relation r_3 to the two-hop compositions. For relations r_1 and r_2, objects are sampled uniformly as in `Uniform`; then, for each subject a, we add the triple (a, r_3, c), where b and c are the entities such that (a, r_1, b) and (b, r_2, c) are in the dataset. This is analogous to real-world relations that have the same multi-hop compositional property (e.g., if a is the mother of b and b is the parent of c, then a is the grandmother of c). This format preserves the property of uniqueness, i.e., for any pair (s, r), o is unique. Step-by-step generation procedures for all dataset families are given in Appendix F (Algorithms 1 and 2).

Power-law Dataset: To probe whether non-uniform fact distributions affect memorization capacity, we sample triples from a Zipf distribution: each triple is assigned a rank $i \in \{1, 2, \ldots, nm\}$ and sampled with probability $p_i \propto i^{-\alpha}$.

4 RESULTS

We characterize how transformer memorization capacity scales with model dimension d (width) and number of layers L (depth).

4.1 WIDTH SCALING: CAPACITY GROWS EXPONENTIALLY WITH THE TRANSFORMER DIMENSION

Figure 1 plots the minimum transformer dimension required for complete memorization across all four dataset families at $L = 1$. Two findings are visible.

Exponential capacity scaling on `Uniform`. As a general observation, across the measured range $N \in [45, 49{,}920]$ the number of non-embedding parameters per memorized fact strictly decreases as the dataset grows; the logarithmic fit below is our best quantitative description of this trend within this range. Table 1 compares candidate fits on the 11 `Uniform` measurements, spanning $N = 45$ to 49,920 (all fits computed in log space; we compare forms with the lack-of-fit F-test for repeated measurements (Draper & Smith, 1998), using the three seeds at each size, Appendix B). The logarithmic-with-offset form $d = 2.756 \ln(N + 80.6) - 10.09$ achieves $R^2 = 0.997$, and is not rejected by a lack-of-fit F-test ($p = 0.67$); the power-law is rejected with or without an offset. The largest dataset sizes are what separate the two forms. With the dataset sizes up to $N = 6{,}240$ only, neither power-law is rejected ($p = 0.23$ without an offset, 0.16 with one); with the sizes up to $N = 12{,}480$, both are ($p = 0.028$ and 0.018).

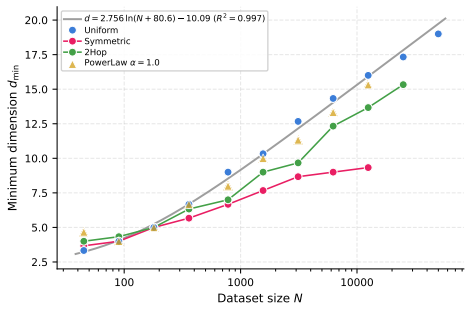


Figure 1: Minimum transformer dimension $d_{\min}$ required for complete memorization vs. dataset size N at $L = 1$, for all four dataset families. **Uniform** (blue circles): the best-fit form is logarithmic-with-offset, $d = 2.756\ \ln(N + 80.6) - 10.09$ ($R^2 = 0.997$); a lack-of-fit F-test rejects the power-law alternatives (Table 1). So non-embedding parameters $P = \Theta(d^2)$ grow as $(\ln N)^2$, which is sublinear in N. **PowerLaw** (amber triangles; Zipf $\alpha = 1.0$): Tested only up to $N = 12{,}480$, since per-batch size grows with the Zipf head-to-tail weight ratio. **Symmetric** (pink circles) and **2Hop** (green circles): datasets with logical structure achieve smaller $d_{\min}$ than the unstructured `Uniform` and `PowerLaw` baselines at every $N \geq 360$, demonstrating that transformers exploit relational regularities to reduce memorization cost.

Table 1: Width-scaling fits on `Uniform` ($L = 1$, 11 measured points, $N = 45$ to 49,920). All fits are computed in log space, with each fit (except the pure logarithmic fit) constrained so that the fitted d is positive for every $N > 0$; the last column is the lack-of-fit F-test p-value ($p < 0.05$ rejects the functional form).

Functional form	**Best fit**	R^2	**Lack-of-fit** p
Logarithmic + offset	$d = 2.756\ \ln(N + 80.6) - 10.09$	0.997	0.67
Logarithmic	$d = 2.17\ \ln N - 5.39$	0.979	9.1×10^{-4}
Power-law + offset	$d = 1.38\ N^{0.259} + 0.00$	0.967	1.7×10^{-5}
Power-law	$d = 1.38\ N^{0.259}$	0.967	3.0×10^{-5}
Linear	$d = 4.82 \times 10^{-4}\ N + 6.30$	0.565	4.4×10^{-16}

Structured datasets require smaller dimension. For `Symmetric` (where the relation is its own inverse) and `2Hop` (where we have two-hop compositional relations), $d_{\min}$ grows more slowly than for `Uniform`. Under our full hyperparameter optimization protocol, this gap is large and consistent from $N = 360$ onward (at the three smallest dataset sizes, all families require similar dimensions); when the hyperparameter search is restricted to the learning rate and the batch size, as in prior empirical work, the three families require the same dimension, as we discuss in the ablation studies (Section 4.3).

4.2 Depth scaling: capacity grows sublinearly with the number of layers

Figure 2 and Table 7 (Appendix E) report the maximum memorizable dataset size $N_{\max}$ as a function of layer count L at fixed width $d = 8$ on `Uniform` data. At fixed width, the maximum memorizable dataset size grows with depth up to $L = 16$ layers, but sublinearly; over the measured range a logarithmic form fits best, but the data do not distinguish between logarithmic and power-law scaling (Table 5, Appendix E).

Two features of the experiment bear on this result. First, the training budget is the same at every depth: 40,000 iterations after warmup, with a hyperparameter search of 60 trials per learning-rate schedule. The reported capacity at each depth comes from three trainings, one per seed, at the largest dataset size that passes. The median iteration at which they converge is 51% of the budget at $L = 1$ and between 77% and 91% at $L \geq 2$. This suggests that at larger values of L a larger budget can raise the measured capacity, which we leave to future work. It's also interesting to note that training is also less stable at larger depth. The fraction of training runs that reach 100% accuracy but do not hold it

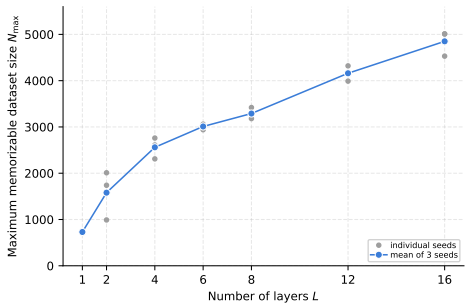


Figure 2: Maximum memorizable dataset size $N_{\max}$ vs. number of layers L at fixed width $d = 8$ (`Uniform` dataset). Grey dots: the three seeds; blue: their mean.

for 50 consecutive steps rises from 2.9% at $L = 1$ to 10.3% at $L = 8$. Second, memorization is a task that one layer can already do at a rate proportional to its parameter count: Nichani et al. (2025) prove that a single layer of self-attention followed by an MLP reaches 100% accuracy on factual recall once the parameter count of either part is linear, up to log factors, in the number of facts, and Chen & Zou (2024) find that one attention layer suffices for memorization while reasoning and generalization need at least two or three.

4.3 Ablation Studies

Impact of suboptimal hyperparameters. To test whether the training protocol alone can account for the difference in scaling with prior work, we ran the width experiment on the same three dataset families and the same architecture while restricting the hyperparameter search to the learning rate and the batch size, the extent of tuning in prior work (Lu et al., 2024; Morris et al., 2026). Under this restricted search the minimum dimension grows as $d \sim N^{0.73\text{–}0.77}$ for all three families (Figure 3, Appendix E), i.e., $P \sim N^{1.5}$: the number of non-embedding parameters grows faster than linearly in N, a scaling even less favorable than the linear relationship reported by prior work, and `Symmetric` and `2Hop` require the same dimension as `Uniform`. Expanding optimization to include weight decay, minimum learning rate, learning rate schedule, and decay duration transforms both findings: Capacity scales exponentially with respect to the transformer dimension for `Uniform` (Figure 1), and the model's ability to leverage structure emerges—structured datasets require fewer parameters than `Uniform` at every dataset size from $N = 360$ on. The optimized configurations consistently achieve perfect memorization with smaller transformer models than any of our earlier experiments.

Batching and maximum training batches. Beyond hyperparameters, two factors proved important. We found that random batch sampling caused loss spikes from adverse batches, requiring many batches to recover and inflating our estimate of the minimum transformer dimension $d_{\min}$; deterministic full-batch updates eliminated this issue (Appendix C). Since each iteration uses the full dataset, each iteration is one exposure per fact, and we use a higher maximum exposure per fact (up to 40,000) than Lu et al. (2024). This is necessary because convergence speed slows heavily as the model dimension approaches $d_{\min}$ (Appendix E, Figure 4). Full-batch training also keeps the exposure per fact the same for every dataset size, whereas with a fixed batch size and number of steps (Morris et al., 2026) it decreases as the dataset grows.

4.4 Generality across architectures: OLMo-3

To test whether our findings are specific to the GPT-2 architecture, we repeated the width scaling experiment on the OLMo-3 architecture at 8 dataset sizes, $N = 45$, 90, 180, 360, 780, 1,560, 3,120, and 6,240, under the same experimental protocol; the OLMo-3 training details are given in Appendix B.1. The results (Table 6, Appendix E) confirm the same scaling behavior: the minimum transformer model dimension grows sublinearly with the dataset size; a lack-of-fit F-test rejects a linear fit ($p = 8.8 \times 10^{-5}$), while a logarithmic fit ($p = 0.68$) and a power-law fit $d \propto N^{0.22}$ ($p = 0.25$) both pass. Interestingly, while the scaling behavior of OLMo-3 is similar to that of GPT-2, OLMo-3 memorizes each dataset at a consistently smaller model dimension.

5 ANALYSIS

Our experiments reveal that memorization capacity grows exponentially with transformer dimension, with structured relations reaching a given capacity at a smaller dimension than unstructured data. For the `Uniform` dataset, the best-fit form for $d_{\min}(N)$ is logarithmic-with-offset, $d = 2.756\,\ln(N + 80.6) - 10.09$ ($R^2 = 0.997$); a lack-of-fit F-test rejects the power-law alternatives (Table 1). Under this fit, non-embedding parameters $P = \Theta(d^2)$ grow as $(\ln N)^2$, which is sublinear in N; equivalently, capacity grows superlinearly in P. Structured datasets (`Symmetric`, `2Hop`) achieve further improvements in efficiency. Over the measured range, the logarithmic fit for $d_{\min}$ implies $P \sim (\ln N)^2$, which grows more slowly than $\sqrt{N}$, the theoretical worst-case bound proved by Kajitsuka & Sato (2025), although within our range P itself is larger than $\sqrt{N}$ (4,617 against 223 at $N = 49{,}920$).

Where the bits live. Table 4 in Appendix E splits the parameters of each model into the embedding table and the non-embedding parameters. The ratio of dataset bits to non-embedding parameters grows with N: from $\sim$1 at N=45 to $\sim$152 at N=49,920. Including embedding parameters, capacity rises from 0.72 bits per parameter at N=45 to $\approx$2.2 bits per parameter at N=49,920 (Table 4). The rate is below 2 at the five smallest sizes ($N \leq 780$) and between 2.04 and 2.19 at every size from $N = 1{,}560$ on, close to the $\sim$2 bits/param reported by Allen-Zhu & Li (2025) for sufficiently trained transformers. A lack-of-fit F-test supports saturation: forms that keep growing are rejected (a logarithm in N, $p = 7.4 \times 10^{-5}$; a power-law, $p = 1.9 \times 10^{-6}$), while the saturating forms $a - b\,N^{-c}$ and $a\,N/(N + k)$ are not ($p = 0.99$ and 0.93) and approach 2.24 and 2.15 bits per parameter.

The embedding table grows with the vocabulary, as in the standard transformer setup. We deliberately do not cap its size below the vocabulary; doing so would insert an artificial bottleneck into the very quantity we seek to measure—the width of the smallest transformer that is able to perfectly memorize the data. We report embedding and non-embedding parameters separately because they play different roles. To test whether the growing vocabulary drives our results, we hold the number of facts fixed at $N = 9{,}216$ and vary the composition of the dataset—from 9,216 entities with 1 relation to 96 entities with 96 relations—shrinking the vocabulary (and with it the embedding table) 48-fold. Every dataset remains fully memorizable, and the minimum transformer model dimension required for perfect memorization ranges from 6 to 46.33, peaking at $m = 64$ before dropping slightly to 45 at $m = 96$. The full experiment is in Appendix E.1.

6 CONCLUSION AND FUTURE WORK

We systematically measured transformer memorization capacity along width and depth axes and find that, under fixed budget, capacity scales exponentially with transformer dimension; and sublinearly with transformer depth. The minimum transformer dimension $d_{\min}$ required to memorize N facts grows logarithmically in N, in contrast to the linear $P \sim N$ scaling reported by prior empirical work. Our results further shed light on aspects of scaling law for memorization capacity that were missing in prior work.

Limitations. Our experiments span $N \in [45, 49{,}920]$. Whether the logarithmic law for $d_{\min}(N)$ extends beyond the measured range is an empirical question. Our measurements use synthetic triples whose information content is exactly known; testing capacity on natural-language datasets such as `bioS` (Allen-Zhu & Li, 2025), where the model must both store the facts and learn to parse and express them in natural language, is an important next step.

Future work. Our scaling experiments vary one axis at a time. Measuring capacity over the joint space of width and depth would help understand the scaling behavior more and recognize how the two axes trade off; we leave this to future work.

AI USE STATEMENT

In this work, we used generative AI tools (large language models) for writing and editing the text of the paper, and for writing research code. We have reviewed all AI-assisted work; every number in the paper was recomputed from the experiment databases by scripts that the authors inspected,

every cited paper was read and its bibliographic entry checked by an author, and AI-written code was audited and tested by the authors before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

All datasets in this work are synthetic (subject, relation, object) triples generated by the procedures in Appendix F. The work involves no human subjects, personal data, or scraped text, and we are not aware of potential conflicts of interest.

Reproducibility statement

The dataset-generation procedures are given in Algorithms 1 and 2; the model and the training protocol in Section 3; the hyperparameter search space, the search procedures, and the search objectives in Appendix B; and the statistical test used to compare scaling forms in Appendix B. The code to reproduce the experiments and the results will be made publicly available once the review process is complete.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: part 3.1, knowledge storage and extraction. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FxNNiUgtfa.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2787–2795, 2013.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium*, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *ICLR*, 2023.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining? In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xingwu Chen and Difan Zou. What can transformer learn with varying depth? case studies on sequence learning tasks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=YNbCbcGyXE.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581.

Norman R. Draper and Harry Smith. *Applied Regression Analysis*. Wiley, April 1998. ISBN 9781118625590. doi: 10.1002/9781118625590. URL http://dx.doi.org/10.1002/9781118625590.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=w7LU2s14kE.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NeurIPS'22, 2022.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pp. 189–206. American Mathematical Society, Providence, RI, 1984.

Takumi Kajitsuka and Issei Sato. Are transformers with one-layer self-attention universal approximators? In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. Single-layer $\Theta(dnN)$ upper bound.

Takumi Kajitsuka and Issei Sato. On the optimal memorization capacity of transformers. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. Tight $\Theta(\sqrt{N})$ bounds.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Junghwan Kim, Michelle Kim, and Barzan Mozafari. Provable memorization capacity of transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8JCg5xJCTPR.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. (mis)fitting scaling laws: A survey of scaling law fitting techniques in deep learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xI71dsS3o4.

Xingyu Lu, Xiaonan Li, Qinyuan Cheng, Kai Ding, Xuanjing Huang, and Xipeng Qiu. Scaling laws for fact memorization of large language models. *arXiv preprint arXiv:2406.15720*, 2024.

Carlo Lucibello and Marc Mézard. The exponential capacity of dense associative memories. *Physical Review Letters*, 132(7):077301, 2024.

Liam Madden, Curtis Fox, and Christos Thrampoulidis. Next-token prediction capacity: General upper bounds and a lower bound for transformers. *IEEE Transactions on Information Theory*, 71 (9):7134–7148, 2025. doi: 10.1109/TIT.2025.3584013.

Sadegh Mahdavi, Renjie Liao, and Christos Thrampoulidis. Memorization capacity of multi-head attention in transformers. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. Originally arXiv:2306.02010, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022.

John Xavier Morris, Chawin Sitawarin, Narine Kokhlikyan, Chuan Guo, G. Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much can language models memorize?, 2026. URL `https://openreview.net/forum?id=NhU661EZ9C`.

Eshaan Nichani, Jason D. Lee, and Alberto Bietti. Understanding factual recall in transformers via associative memories. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=hwSmPOAmhk`.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *EMNLP-IJCNLP*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021.

Shauli Ravfogel, Gilad Yehudai, Joan Bruna, and Alberto Bietti. Geometric factual recall in transformers, 2026. URL `https://arxiv.org/abs/2605.12426`.

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

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=ByxRM0Ntvr`.

A Detailed Optimization Formulation

We measure transformer memorization capacity in two complementary ways: in width experiments, we find the minimum transformer dimension d^* for a fixed dataset; in depth experiments, we find the maximum dataset size N^* for a fixed transformer. Both depend critically on training hyperparameters: suboptimal learning rates, weight decay, or schedules can prevent memorization even when the model has sufficient capacity.

Let θ denote the hyperparameter vector listed in Table 2. Let $\text{Train}(d, L, N; \theta)$ denote training a transformer of dimension d and depth L on a dataset of size N with hyperparameters θ until either reaching 100% accuracy or exhausting the training budget.

Width experiments ($L = 1$, dataset size N fixed).

$$d^*(\theta; N) = \min\{\, d \in \mathbb{N} : \text{Train}(d, 1, N; \theta) \text{ achieves } 100\% \text{ accuracy} \,\}. \tag{1}$$

We seek $\theta^*(N) = \text{argmin}_\theta\, d^*(\theta; N)$ for each N in our experimental sweep.

Depth experiments (dimension d and number of layers L fixed).

$$N^*(\theta; d, L) = \max\{\, N \in \mathbb{N} : \text{Train}(d, L, N; \theta) \text{ achieves } 100\% \text{ accuracy} \,\}. \tag{2}$$

We seek $\theta^*(d, L) = \text{argmax}_\theta\, N^*(\theta; d, L)$ for each L.

Procedural details (per-trial search, multi-seed evaluation, BO objective formula) are in Appendix B.

B Hyperparameter Search Procedure

Table 2: Hyperparameter search space. Decay schedule is treated as a stratification variable: a separate optimization study is run for each schedule type and the best is reported. Warmup is fixed (not searched), as initial optimization results showed this would not hurt optimal value achieved. Discussed in Section 3.

Parameter	Search space	Notes
Learning rate	$[10^{-6}, 1]$ (log)	depth: $[10^{-7}, 3]$
Use weight decay	{False, True}	gate; if False, wd $= 0$
Weight decay	$[10^{-8}, 10^{-1}]$ (log)	only sampled when gate is True
Min-LR ratio	$[10^{-8}, 1.0]$ (log)	min_lr $=$ lr $\cdot$ ratio
Schedule-shape:		
Decay end step	[100, 40,000]	step (after warmup) where decay ends
LR decay schedule	{`linear`, `cosine`}	stratified (separate study per type)
Warmup steps	$500 \cdot L$	fixed; depth: $\min(500 \cdot L, 4000)$

Note that log on the table means the optimization was done in the log space.

Per-trial dimension search (width experiments, Algorithm 3). For each candidate configuration θ and seed s, we locate $d_s^*(\theta)$ as follows:

1. *Geometric halving downward.* Start at $d = 32$ and halve ($d = 32, 16, 8, \ldots$), training at each d and recording the smallest passing dimension.
2. *Upward fallback.* If $d = 32$ itself fails, double upward ($d = 64, 128, 256$) until a passing dimension is found. If none passes by $d = 256$, the seed is marked as failed.
3. *Binary search.* Refine between the last passing and first failing dimension to pinpoint d_s^*.

Per-trial dataset-size search (depth experiments, Algorithm 4). For each candidate configuration θ and seed s, we locate $N_s^*(\theta)$ as follows:

1. *Geometric doubling/halving.* Starting from an initial estimate of N, double or halve the dataset size until the first success–failure boundary is found.
2. *Binary search.* Refine between the last passing and first failing dataset size to pinpoint N_s^*.

Multi-seed objective. Each candidate configuration is evaluated jointly on three seeds (10, 20, 30) run in parallel across available GPUs, and is scored by

$$\text{objective} = \min_i d_i + 0.1\, n_{\text{failed}} + 0.05 \sum_i \mathbf{1}\{d_i < \infty\}\,(d_i - \min_j d_j), \tag{3}$$

where d_i is the smallest transformer model dimension at which the training run with seed i memorizes the dataset, $d_i = \infty$ if seed i fails to memorize within the training budget, and $n_{\text{failed}} = \#\{i : d_i = \infty\}$ is the number of failed seeds. The objective primarily minimizes the transformer model dimension; the two remaining terms serve only as tiebreakers between hyperparameter configurations that reach the same minimum dimension, preferring configurations with more reliable training (fewer failed seeds and less spread across seeds). The coefficients 0.1 and 0.05 are chosen small enough that the tiebreakers cannot outweigh the minimum dimension: because the dimension is an integer, changing the winning configuration would require the tiebreaker terms to add up to at least 1, which in turn would require the seeds to disagree by about 20 dimensions in total. No completed configuration came close to this.

Multi-seed objective (depth experiments). In the depth experiments, each seed runs its own dataset-size search, and each hyperparameter configuration is scored by

$$\text{objective} = \max_i N_i - 0.1\, n_{\text{failed}}\, S - 0.05 \sum_i \mathbf{1}\{N_i > 0\}\,(\max_j N_j - N_i), \tag{4}$$

where N_i is the largest dataset size that the training run with seed i memorizes, $N_i = 0$ if seed i memorizes no dataset size, $n_{\text{failed}} = \#\{i : N_i = 0\}$ is the number of failed seeds, and $S = 1{,}002$ facts is the coarse step of the size search. The objective primarily maximizes the dataset size; the two remaining terms prefer configurations with more reliable training (fewer failed seeds and less spread across seeds). Unlike in the width experiments, these terms are measured in facts and can change which configuration scores highest. The objective only guides the search: for each depth we report the configuration with the largest mean over the three seeds (Table 7), which does not use the objective.

Lack-of-fit F-test. To test whether a functional form f with k parameters describes a scaling curve measured at m points $x_1, \ldots, x_m$ (dataset sizes, or numbers of layers), we use the lack-of-fit F-test for repeated measurements (Draper & Smith, 1998). At each point we have three values y_{i1}, y_{i2}, y_{i3}, one per seed, where y is the logarithm of the measured quantity ($d_{\min}$, $N_{\max}$, or bits per parameter). Let $\bar{y}_i$ be their mean. We fit $\ln f$ to the means $\bar{y}_i$ by least squares, and compute the pure-error and lack-of-fit sums of squares

$$\text{SS}_{\text{PE}} = \sum_{i=1}^{m}\sum_{j=1}^{3}(y_{ij} - \bar{y}_i)^2, \qquad \text{SS}_{\text{LOF}} = 3\sum_{i=1}^{m}\big(\bar{y}_i - \ln f(x_i)\big)^2,$$

with $2m$ and $m - k$ degrees of freedom. If f is the correct form, $F = \frac{\text{SS}_{\text{LOF}}/(m-k)}{\text{SS}_{\text{PE}}/(2m)}$ follows an $F(m-k,\, 2m)$ distribution; we report $p = \Pr[F(m-k, 2m) > F]$ and reject the form when $p < 0.05$. The test asks whether the fitted curve misses the means by more than the spread across seeds explains, so a large p means the form is consistent with the data, not that it is proven.

B.1 Generality Across Architectures: OLMo-3 Implementation and Results

To evaluate whether logarithmic width scaling is specific to GPT-2 or generalizes across modern transformer architectures, we repeat the width scaling experiment using the OLMo-3 architecture.

To maintain parameter fairness against our GPT-2 baseline (Section 3.2) at width d, the OLMo-3 block is aligned via three adjustments:

1. Disabling Positional Embeddings for position-free processing across all integer widths d;
2. Using standard full attention matching GPT-2; and
3. Setting the SwiGLU (Swish-Gated Linear Unit) feed-forward hidden dimension to $\lfloor 8d/3 \rfloor$ so that its 3-matrix FFN parameter count matches GPT-2's standard 2-matrix FFN up to rounding.

Models are trained using full-batch gradient descent, bfloat16 mixed precision, and the Adam optimizer. For each dataset size N, hyperparameters are optimized over the search space of Table 2 with 60 Optuna TPE trials per learning-rate schedule. Convergence requires maintaining 100% training accuracy for 50 consecutive evaluation steps within a budget of 40,000 batches.

As detailed in Table 3, the minimum required dimension $d_{\min}$ for OLMo-3 grows from $d = 2$ at $N = 45$ to $d = 6$ at $N = 6{,}240$; as for GPT-2, a lack-of-fit F-test rejects a linear fit ($p = 8.8 \times 10^{-5}$) while a logarithm ($p = 0.68$) and a power law with exponent 0.22 ($p = 0.25$) both pass.

Table 3: OLMo-3 factual capacity measurements across dataset sizes N. $d_{\min}$ is the minimum width achieving 100% memorization accuracy across three random seeds (seeds 10, 20, 30).

Entities (n)	**Facts** (N)	**Information (Bits)**	**Seeds (10 / 20 / 30)** $d_{\min}$	**Mean** $d_{\min}$	**Best** $d_{\min}$
15	45	176	2 / 2 / 2	2.00	**2**
30	90	442	3 / 4 / 2	3.00	**2**
60	180	1,063	3 / 3 / 3	3.00	**3**
120	360	2,486	4 / 4 / 4	4.00	**4**
260	780	6,257	4 / 4 / 5	4.33	**4**
520	1,560	14,075	5 / 5 / 5	5.00	**5**
1040	3,120	31,270	6 / 6 / 6	6.00	**6**
2080	6,240	68,780	6 / 6 / 6	6.00	**6**

C Sampling Sensitivity Analysis

Throughout our experiments, we found that memorization—and, consequently, the estimated minimum transformer dimension d^*—was highly sensitive to *random sampling* of training examples. This sensitivity persisted even when the sample size used per update exceeded the dataset size by factors (random sampling with replacement), indicating that the issue was not whether the sample was small or large, but rather the randomness of the draw itself. In particular, we observed that a single adverse sample (a "bad batch" in the sense of an unlucky random draw) could induce a large, misaligned update that slowed loss descent, as evidenced by spikes or plateaus in the training loss traces (Figure 5). When we removed this source of noise by using *deterministic full-dataset updates* (i.e., the exact empirical gradient computed over a fixed example order), the minimum feasible dimension dropped significantly, and the region of settings that achieved this minimum became stable across seeds.

Based on this discovery, for all subsequent experiments, we fix the batch size: for `Uniform`, `Symmetric`, and `2Hop` datasets, we set $b = N$ (the full dataset size), processing all triples in each batch. For `PowerLaw` distributions, we use Power-Law sampling as mentioned in Appendix D.

D Power-Law Sampling

To probe whether non-uniform fact distributions affect memorization capacity, we sample triples according to a Zipf distribution. For a dataset with n entities and $m = 3$ relations (nm total triples), each triple is assigned a rank $i = 1, 2, \ldots, nm$ and sampled with probability:

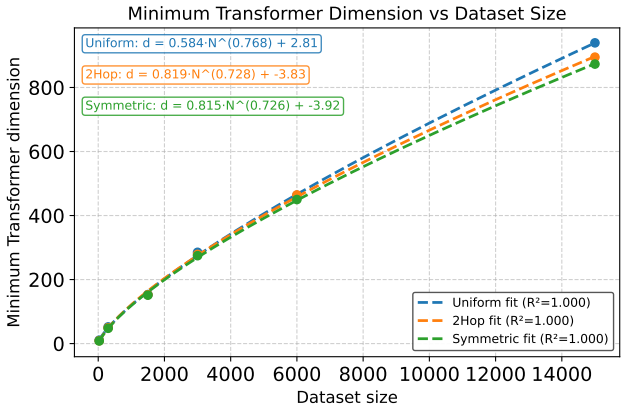
$$p_i = \frac{i^{-\alpha}}{Z}, \quad \text{where} \quad Z = \sum_{j=1}^{nm} j^{-\alpha} \tag{5}$$

and we use $\alpha = 1.0$ in our experiments.

The replication rule is: each fact i appears $\max(1, \lceil k\, p_i \rceil)$ times per batch, where the multiplier $k = 1/p_N$ is set so that the rarest fact (smallest p_i) appears exactly once per batch. The batch is constructed once and reused for every training iteration (Algorithm 2). The resulting per-batch size grows roughly as k, which is why we capped tested N at $12{,}480$ (Section 3.6).

E Additional Experimental Results

Positional embeddings. In preliminary experiments we trained models with and without learned positional embeddings. Since every input has the fixed form (s, r), position is implicit in the input structure, and positional embeddings did not change the minimum transformer model dimension required for memorization; we therefore omit them in all experiments.



(a) Limited Hyperparameter optimization

Figure 3: Minimum model dimension required for complete memorization vs. dataset size with power-law fit (offset unconstrained) across dataset types (`Uniform`, `Symmetric`, `2Hop`), when we only optimized learning rate and batch size. The minimum transformer dimension required for complete memorization is way higher than under the full search (Figure 1), highlighting the impact of hyperparameter optimization.

Table 4: Non-embedding-vs-embedding parameter breakdown for the smallest single-layer model that memorizes a `Uniform` dataset of size N ($d_{\min}$ averaged over 3 seeds). Non-embedding: $P = 12d^2 + 15d$; embedding: $P_{\text{emb}} = (E + R) \cdot d$; bits $= N \log_2 E$. Including embedding parameters, capacity rises from 0.72 to $\approx$2.2 bits per parameter, reaching $\geq$2.0 from $N = 1{,}560$ onward. Discussed in Section 5.

N	$d_{\min}$	Body %	bits/P_{body}	bits/P_{tot}
45	3.33	75%	1.0	0.72
90	4.00	66%	1.8	1.15
180	5.00	54%	2.8	1.54
360	6.67	44%	3.9	1.71
780	9.00	32%	5.7	1.80
1,560	10.33	21%	9.8	2.06
3,120	12.67	14%	14.8	2.04
6,240	14.33	8%	25.7	2.11
12,480	16.00	5%	45.3	2.15
24,960	17.33	3%	84.1	2.19
49,920	19.00	1%	151.6	2.18

Table 5: Depth-scaling fits on `Uniform` (fixed width $d = 8$, 7 depths, 3 seeds). All fits are computed in log space; the last column is the lack-of-fit F-test p-value ($p < 0.05$ rejects the functional form).

Functional form	**Best fit**	R^2	**Lack-of-fit** p
Proportional	$N = 498\,L$	0.690	7.8×10^{-6}
Linear with intercept	$N = 338\,L + 597$	0.906	0.0025
Power-law, exponent ≥ 1	$N = 498\,L^{1.000}$	0.690	4.0×10^{-6}
Exponential	$N = 1197\,e^{0.103\,L}$	0.741	1.2×10^{-5}
Logarithmic	$N = 1331\,\ln L + 709$	0.992	0.73
Power-law	$N = 870\,L^{0.652}$	0.966	0.092

Table 6: OLMo-3 width scaling experiment: minimum transformer model dimension d needed to memorize all N facts, with $m = 3$ relations (averaged over 3 seeds; GPT-2 shown for comparison).

N	45	90	180	360	780	1,560	3,120	6,240
OLMo-3	2.00	3.00	3.00	4.00	4.33	5.00	6.00	6.00
GPT-2	3.33	4.00	5.00	6.67	9.00	10.33	12.67	14.33

Table 7: Depth scaling experiment at transformer model dimension 8: maximum dataset size memorized completely, for each number of layers L (`Uniform`, $m = 3$). Each entry is the mean over seeds 10, 20, 30 for the hyperparameter configuration with the largest mean; the range is the smallest and largest seed.

layers L	1	2	4	6	8	12	16
max N (mean)	730	1,580	2,560	3,010	3,290	4,160	4,850
smallest seed	720	990	2,310	2,940	3,180	3,990	4,530
largest seed	750	2,010	2,760	3,060	3,420	4,320	5,010

E.1 VARYING THE NUMBER OF RELATIONS AT FIXED DATASET SIZE

To test whether the growing vocabulary drives our results, we hold the number of facts fixed at $N = 9{,}216$ and vary the composition of the dataset—from 9,216 entities with 1 relation to 96 entities with 96 relations—shrinking the vocabulary (and with it the embedding table) 48-fold. Every dataset remains fully memorizable: the minimum transformer model dimension required for perfect memorization ranges from 6 to 46.33, peaking at $m = 64$ before dropping slightly to 45 at $m = 96$. A joint fit with the width scaling experiment's dataset sizes gives $d \approx 2.58 \ln n + 12.65 \ln m - 18.71$ ($R^2 = 0.973$), where n is the number of entities and m the number of relations; a lack-of-fit F-test rejects this form ($p < 10^{-10}$).

Table 8: Minimum transformer model dimension d needed to memorize $N = 9{,}216$ facts as we vary the composition of the dataset, from 9,216 entities with 1 relation to 96 entities with 96 relations (averaged over 3 seeds).

entities n	9216	4608	3072	2304	1152	576	288	144	96
relations m	1	2	3	4	8	16	32	64	96
min. dimension d to memorize all facts	6.00	10.67	15.33	19.00	29.33	38.00	44.00	46.33	45.00

Figure 4 illustrates the relationship between the number of batches required for 100% training accuracy vs the transformer dimension d across different dataset sizes. We observe a clear "capacity compute tradeoff": a smaller dimension requires many more training batches to achieve the same level of capacity as larger models.

Furthermore, the plot compares 2 different decay types and shows that both schedules eventually converge to a similar minimum dimension. This figure shows that a high number of batches is essential for uncovering the true scaling regime between transformer dimension and dataset size.

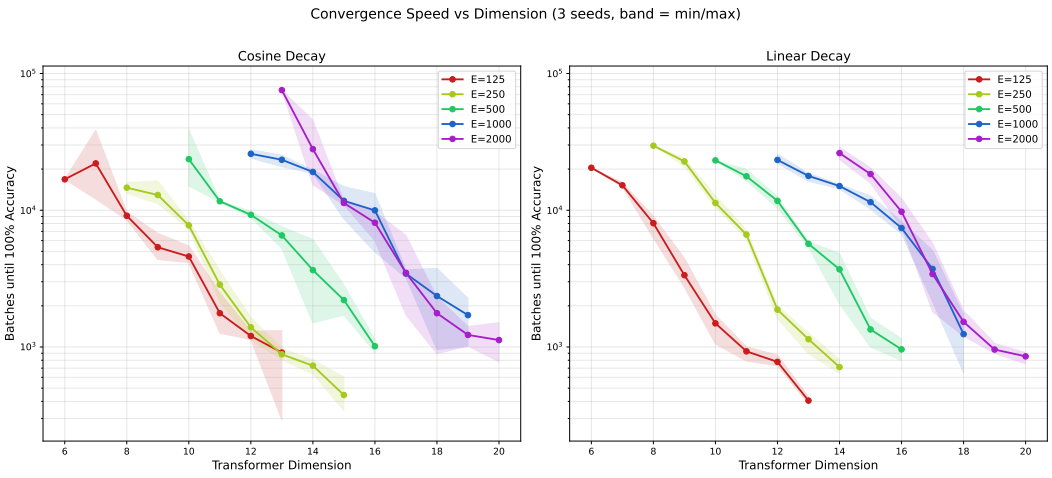


Figure 4: **Convergence Speed vs. Dimension.** The total number of batches required to achieve 100% training accuracy vs transformer dimension d for different entity sizes $E \in \{125, 250, 500, 1000, 2000\}$. Plots for both the linear and cosine decay types are shown. The y-axis is on a log scale, showing that as the model approaches the theoretical minimum dimension d_{min}, the cost required to train for perfect memorization increases exponentially. The shaded bands represent the variation of the number of batches required over 3 seeds.

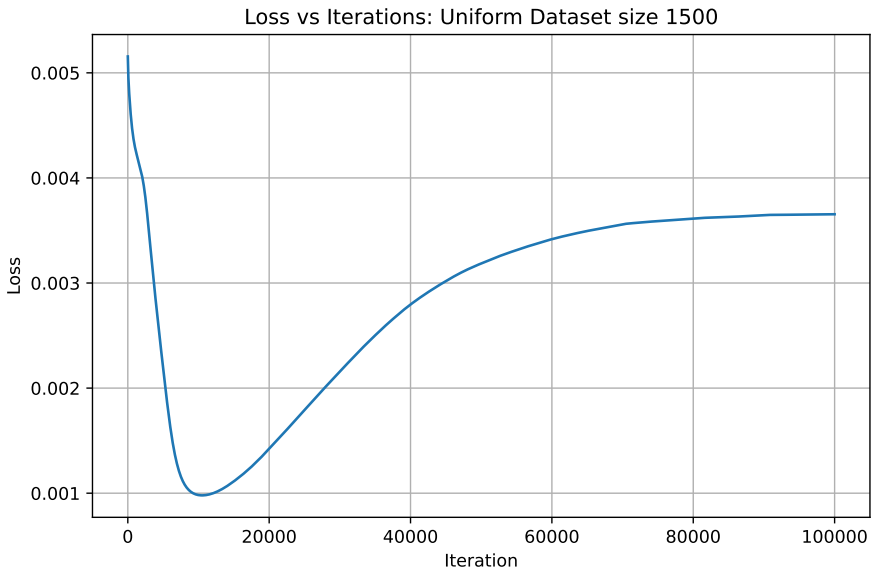


Figure 5: Training loss vs. number of iterations under random batch sampling (`Uniform` dataset of size 1500, transformer dimension 95). Sudden spikes correspond to "bad batches"—unlucky random draws whose gradient is misaligned with the loss surface, causing transient divergence and slowing convergence. These episodes inflate measured d^* when the iteration cap is finite. Switching to deterministic full-batch updates eliminates them and reduces the minimum dimension required for memorization.

F Experimental Algorithms

Algorithm 1 Dataset generation: GenerateDataset(n, m, $type$)

Require: number of entities n, number of relations m, $type \in \{\texttt{Uniform}, \texttt{Symmetric}, \texttt{2Hop}\}$. Each entity and relation is a single vocabulary token ($|V| = n + m$). For `2Hop`, $m = 3$ and the last relation r_3 is dedicated to the two-hop compositions.

$D \leftarrow \emptyset$
for all relations $r \in \{1, \ldots, m\}$ **do**
  for all subjects $s \in \{1, \ldots, n\}$ **do**
    if $type = \texttt{Uniform}$ **then**
      sample o uniformly from $\{1, \ldots, n\}$; add (s, r, o) to D
    else if $type = \texttt{Symmetric}$ **then**
      if there does not exist o' such that $(o', r, s) \in D$ **then**
        $O' \leftarrow \{o : (s', r, o) \notin D$ and $(o, r, s') \notin D$ for any entity $s'\}$
        sample o uniformly from O'; add (s, r, o) and its mirror (o, r, s) to D (a single fact if $o = s$)
      end if
    else if $type = \texttt{2Hop}$ **and** $r \in \{r_1, r_2\}$ **then**
      sample o uniformly from $\{1, \ldots, n\}$; add (s, r, o) to D
    else if $type = \texttt{2Hop}$ **and** $r = r_3$ **then**
      add (s, r_3, c) to D, where b and c are the entities such that $(s, r_1, b) \in D$ and $(b, r_2, c) \in D$ (these facts exist because r_3 is processed last)
    end if
  end for
end for
return D {every (subject, relation) pair appears exactly once: $N = n \cdot m$ facts, each a 3-token sequence; the training loss is computed on the object token only}

Algorithm 2 Batch construction for `PowerLaw`: ConstructBatch(D, α)

Require: dataset D of N facts, Zipf exponent $\alpha > 0$. The batch is constructed once and reused for every training iteration.

assign each fact $i \in D$ a Zipf probability $p_i \propto i^{-\alpha}$ (fixed for the dataset; they do not change across batches)
$k \leftarrow 1/p_N$, where p_N is the probability of the rarest fact {the rarest fact appears exactly once per batch}
construct the batch so that each fact i appears $\max(1, \lceil k\, p_i \rceil)$ times
return the batch {for `Uniform`, `Symmetric`, and `2Hop`, each batch contains each fact exactly once}

Algorithm 3 Per-seed width search: MinDim(N, θ, s)

Require: dataset size N, hyperparameters θ, seed s; start $d_0 = 32$, cap $d_{\max} = 256$. TRAIN(d) returns pass if $\mathrm{Train}(d, 1, N; \theta)$ with seed s reaches stable 100% accuracy.
Ensure: $d_s^*(\theta)$, or ∞ if no $d \le d_{\max}$ passes

$d \leftarrow d_0$; $d_{\text{ok}} \leftarrow \infty$
while $d \geq 1$ **and** TRAIN(d) passes **do**
  $d_{\text{ok}} \leftarrow d$; $d \leftarrow \lfloor d/2 \rfloor$ {geometric halving}
end while
if $d_{\text{ok}} = \infty$ **then**
  $d \leftarrow 2d_0$
  while $d \le d_{\max}$ **and** TRAIN(d) fails **do**
    $d \leftarrow 2d$ {upward fallback}
  end while
  if $d > d_{\max}$ **then**
    return ∞
  end if
  $d_{\text{ok}} \leftarrow d$; $d_{\text{fail}} \leftarrow d/2$
else
  $d_{\text{fail}} \leftarrow d$ {first failing dimension (0 if every $d \geq 1$ passed)}
end if
$lo \leftarrow d_{\text{fail}} + 1$; $hi \leftarrow d_{\text{ok}} - 1$
while $lo \le hi$ **do**
  $m \leftarrow \lfloor (lo + hi)/2 \rfloor$
  if TRAIN(m) passes **then**
    $d_{\text{ok}} \leftarrow m$; $hi \leftarrow m - 1$
  else
    $lo \leftarrow m + 1$
  end if
end while
return d_{ok}

Algorithm 4 Per-seed dataset-size search: MaxSize(d, L, θ, s)

Require: width d, depth L, hyperparameters θ, seed s; initial estimate N_0; coarse step $S = 1{,}002$ facts; fine step $F = 30$ facts. TRAIN(N) returns pass if $\mathrm{Train}(d, L, N; \theta)$ with seed s reaches stable 100% accuracy. $\mathrm{round}_S(\cdot)$ rounds to the nearest multiple of S, never below S; $\mathrm{round}_F(\cdot)$ likewise for F.

Ensure: $N_s^*(\theta)$, or 0 if no size passes

if TRAIN(N_0) passes **then**
  $N_{\text{ok}} \leftarrow N_0$; $N \leftarrow \mathrm{round}_S(2N_0)$
  while TRAIN(N) passes **do**
    $N_{\text{ok}} \leftarrow N$; $N \leftarrow \mathrm{round}_S(2N)$
  end while{doubling}
  $N_{\text{fail}} \leftarrow N$
else
  $N_{\text{fail}} \leftarrow N_0$; $N \leftarrow \mathrm{round}_S(N_0/2)$
  while TRAIN(N) fails **do**
    $N_{\text{fail}} \leftarrow N$
    if $N/2 < F$ **then**
      return 0
    end if
    $N \leftarrow \mathrm{round}_S(N/2)$ if $N/2 \geq S$, else $\mathrm{round}_F(N/2)$ {halving: coarse grid above S, fine grid below}
  end while
  $N_{\text{ok}} \leftarrow N$
end if
$lo \leftarrow$ the smallest multiple of F above N_{ok}; $hi \leftarrow$ the largest multiple of F below N_{fail} {binary search on the F grid}
while $lo \leq hi$ **do**
  $m \leftarrow$ the multiple of F nearest to $(lo + hi)/2$
  if TRAIN(m) passes **then**
    $N_{\text{ok}} \leftarrow m$; $lo \leftarrow m + F$
  else
    $hi \leftarrow m - F$
  end if
end while
return N_{ok}