SCALING LAWS FOR ASSOCIATIVE MEMORIES

Vivien Cabannes
FAIR, Meta

Elvis Dohmatob
FAIR, Meta

Alberto Bietti
Flatiron Institute

ABSTRACT

Learning arguably involves the discovery and memorization of abstract rules. The aim of this paper is to study associative memory mechanisms. Our model is based on high-dimensional matrices consisting of outer products of embeddings, which relates to the inner layers of transformer language models. We derive precise scaling laws with respect to sample size and parameter size, and discuss the statistical efficiency of different estimators, including optimization-based algorithms. We provide extensive numerical experiments to validate and interpret theoretical results, including fine-grained visualizations of the stored memory associations.

1 INTRODUCTION

As the scale of large language models (LLMs) keeps increasing, scaling laws have become a crucial tool to empirically assess and predict the behavior of these models when varying the number of parameters and training data (Kaplan et al., 2020; Hoffmann et al., 2022). Despite their practical impact, the underlying phenomena leading to such scaling laws remain poorly understood. A better understanding of such phenomena could guide researchers towards improved models, algorithms, and datasets which may lead to improved scaling laws.

Our study focuses on a simple model that aims to be representative of LLMs in two ways. First, we focus on heavy-tailed data distributions over discrete tokens, a natural assumption for text data (Piantadosi, 2014). Second, we consider associative memory models that store input-output pairs through outer-products of finite-dimensional embeddings, and can be seen as a proxy of the intermediate layers of transformers. Indeed, some transformer layers have been found to behave as key-value memories (Geva et al., 2021; Meng et al., 2022), and more generally outer-product associative memory matrices arise naturally from training dynamics on intermediate weights (Bietti et al., 2023). Beyond simple associative recall, the combination of multiple such associative rules at different layers may lead to certain circuits with rich "reasoning" behaviors based on context (Elhage et al., 2021; Bietti et al., 2023; Michaud et al., 2023). For example, an intermediate layer input token may encode for the topic "linux", leading to an output token that will trigger a specific behavior in the transformer's following layers when processing the token "terminal".

Our contributions are as follows:

- We provide precise statistical rates for outer-product memories with random embeddings, and compare different memory storage schemes in the context of Zipf-distributed data.
- We compare theoretical schemes to the weights learned by various optimization algorithms used in practice, and illustrate the role of different design choices with numerical experiments.

Related work. Associative memory models have a long history in the literature on neural computation (Steinbuch, 1961; Willshaw et al., 1969; Longuet-Higgins et al., 1970; Kohonen, 1972; Amari, 1972; Little, 1974; Hopfield, 1982; Smolensky, 1990; Schlag et al., 2021; Valle-Lisboa et al., 2023), though the statistical insights we provide for continuous-values random embeddings and heavy-tailed tokens distributions are new, to the best of our knowledge. Memorization behaviors have drawn a lot of attention recently, and are believed to be an important notion to understand the learning happening in deep neural network (e.g., Sukhbaatar et al., 2019; Feldman, 2020; Feldman & Zhang, 2020; Geva et al., 2021; Wu et al., 2022). Building on memorization and heavy-tailed discrete data, our model bears similarities to the ones of Hutter (2021), Michaud et al. (2023) or Debowski (2023), although we focus on practical models with finite capacity. The discrete nature of

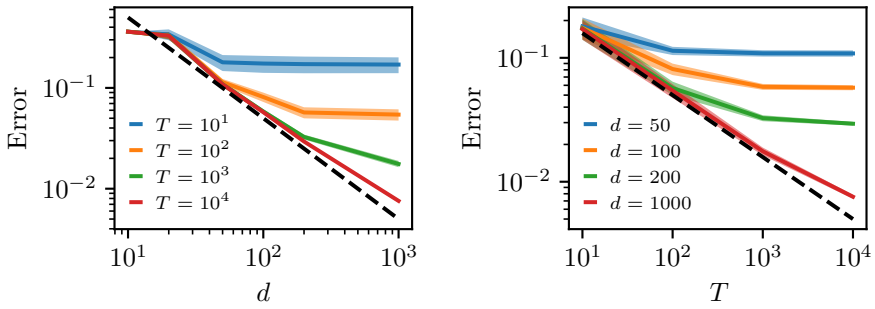

Figure 1: Scaling laws with respect to model capacity d (left), respectively the number of data seen T (right), for various numbers of dataset size T, respectively various model capacity d. This plots validates empirically the theory developed in the paper that proves scaling laws in $\mathcal{E}(f_q) \asymp d^{-\alpha+1} + T^{-1+1/\alpha}$ (dashed lines) under our setting with $\alpha = 2$ (1), (2), (5), and the association scheme (12) with $\rho = 0$ and $P = d/8$. The experiments averaged over 100 runs, standard deviations are shown with solid color.

tokens contrasts with other recent works on scaling laws that have focused on continuous Gaussian inputs (e.g., Bahri et al., 2021; Maloney et al., 2022; Sorscher et al., 2022).

2 Model for Associative Memory

The data. In the following, we consider a joint distribution $p \in \Delta_{[N]\times[M]}$ on inputs $x \in [N]$ and outputs $y \in [M]$. The inputs and outputs are respectively assumed to solely take N and M discrete values respectively. For example, N could be the number of potential sequences of fixed word length in the English language, while M would be all the potential words to complete the sequence. Abstractly, x and y will be referred to as tokens. To simplify the study, we assume for now that y is a deterministic function of x, i.e., there is no noise in the labels. In consistency with language modeling, we equally assume that $p(x)$ follows a Zipf law. Formally, there exists an parameter $\alpha > 0$, a normalizing constant C_α, a permutation $\sigma \in \mathfrak{S}_n$ and a function $f_* : [N] \to [M]$ such that

$$\forall\, x, y \in [N] \times [M], \qquad p(\sigma(x)) = C_\alpha x^{-\alpha}, \qquad p(y|x) = \mathbf{1}_{y=f_*(x)}. \tag{1}$$

The distribution p is not known, but has generated T known independent samples $(x_t, y_t)_{t\in[T]} \sim p$. For readability sake, we will assume without restriction that σ is the identity (so that p is decreasing).

The model, and the loss. The input tokens are embedded into a space $\mathbb{R}^d$ of dimension d through an embedding map $e : [N] \to \mathbb{R}^d$. This space is used for computation purposes. In particular, we focus on the linear transformation parameterized by a matrix $W \in \mathbb{R}^{d\times d}$ mapping x to $We(x)$. This latter vector is mapped back to the output space through an unembedding map $u : [M] \to \mathbb{R}^d$ and the decoding rule

$$f_W(x) = \arg\max_{y\in[M]} u_y^\top W e_x, \qquad W \in \mathbb{R}^{d\times d}, \tag{2}$$

where e_x and u_y are abbreviations for $e(x)$ and $u(y)$. The model (2) can be seen as analogous to an attention layer where keys e_x are tested against queries u_y through a matrix W before going through a softmax layer, which, when the attention is peaky, identifies to an argmax. It also resembles next-token prediction from an intermediate representation We_x, which may itself be the output of an attention block that attends to a token x. The matrices W will be expressed as associative memories. Memory of an observed pair (x, y) is represented as an outer product $u_y e_x^\top$. Remembering those with respect to a probability $q \in \Delta_{[N]\times[M]}$ leads to the matrix

$$W_q = \sum_{(x,y)\in[N]\times[M]} q(x, y) u_y e_x^\top, \qquad q \in \Delta_{[N]\times[M]}, \tag{3}$$

This representation (3) is justified as the predictions (2) are insensitive to modifications of M outside the span of $(u_y e_x^\top)_{x,y}$. In our deterministic setting (1) where one only observes pairs $(x, f_*(x))$, we shall consider the simpler model where[1]

$$W_q = \sum_{x\in[N]} q(x) u_{f_*(x)} e_x^\top, \qquad q \in \Delta_{[N]}. \tag{4}$$

[1]It should be noted that the proof techniques behind Theorem 1 do not break when considering $q = q(x, y)$: both models would lead to similar results, with the case $q = q(x, y)$ being simpler to comprehend.

Table 1: Summary of key elements in the study. We are given discrete tokens x, y with deterministic relation $y = f_*(x)$. We embed tokens in $\mathbb{R}^d$, d acts as a "model capacity" parameter. We store association $x \to y$ in the matrix W through a scheme q and recall them through the decoding f_q. We will first study the scaling law of the generalization error $\mathcal{E}$ as a function of the number of data T, and the model capacity d for different schemes q. We will later study the scheme q found by optimization-based algorithms.

Tokens	Embeddings	Model	Scaling
$y_{i_t} = f_*(x_{i_t})$	$e_x, u_y \in \mathbb{R}^d$	$W = \sum_x q(x) u_{f_*(x)} e_x^\top$	$\mathcal{E}(q) = \mathbb{E}[\mathbf{1}_{f_q(x) \neq f_*(x)}]$
$t \in \{1, 2, \dots, T\}$	$e_x \sim \mathcal{N}(0, I)$	$f_q(x) = \arg\max_y u_y W e_x$	$\mathcal{E}(q) = F(d, T; q)$

Table 2: Some insightful provable scaling laws with respect to the model capacity d, and the number of data T, for two schemes that store associations as (4) and random embeddings.

Model	Error scaling	Comment
$q(x) = p(x)$	$d^{-(\alpha-1)/2\alpha} + T^{-1+1/\alpha}$	Found with large batches in one step
$q(x) = \mathbf{1}_{x \leq d}$	$d^{-\alpha+1} + T^{-1+1/\alpha}$	Optimal scaling with random embeddings

To simplify notations, we will write f_q for f_{W_q} (2). The model f_q is seen as superposing memories since all associations are mixed together in a single matrix. The quality of a mapping f is quantified through the generalization error

$$\mathcal{E}(f) = \mathbb{E}_{(X,Y)\sim p}[\mathbf{1}_{f(X)\neq Y}], \qquad f : [N] \to [M]. \tag{5}$$

Which questions are we interested in? Several questions naturally arise from our model. The first ones are related to scaling laws: how does the error depend on T, the number of data? How does it scale with d that encodes for model capacity? The second ones relate to the model itself: how does the error behave for different q? What about optimization-based algorithms?

Arguably, the model (2) lays out a simple model to study memorization, which could easily be extended to model more intricate memorization and training behaviors inside a transformer language model. Indeed, memories of the form (4) were found to accurately model the behavior of weight matrices in multi-layer transformers trained by gradient methods on certain tasks (Bietti et al., 2023). Hence, we expect our study to be generalizable to more complex mechanisms in transformers, resulting in rich token interactions to predict the next token in a sequence.

3 Scaling laws with random embeddings

Why do we make errors? With a simple deterministic model, one may wonder how can we not learn perfectly the mapping f_*. There are two sources of error. One is due to not having enough data to see all the potential association $(x, f_*(x))$, and has already been studied by Hutter (2021). The other one is due to the limited memory capacity of our model, which we illustrate in Figure 2.

Proposition 1 (Finite data, infinite memory)**.** *Consider a infinite memory model $\hat{f}$, which at time T predicts correctly all x that were seen in the past training, i.e., $x \in \{X_t\}_{t\in[T]}$, where the (X_t, Y_t) were drawn independently at random from a distribution $p \in \Delta_{[N]\times[M]}$. Under the data model the generalization error reads, with respect to the random dataset $\mathcal{D}_T = (X_t, Y_t)_{t\in[T]}$,*

$$\mathbb{E}_{\mathcal{D}_T}[\mathcal{E}(\hat{f})] \asymp T^{-1+1/\alpha}. \tag{6}$$

Here, the notation $a \asymp b$ means that there exist two constants c_1 and c_2 such that $c_1 b \leq a \leq c_2 b$.

3.1 Tight error characterization

The case where one has infinite data but finite memory is intrinsically a deterministic problem. However, characterizing interferences between embeddings and the corresponding generalization error is combinatorial in nature, and is hard to study without specific assumptions on the embeddings e and u. A natural choice is to consider them to be random, as is the case at initialization.

Theorem 1 (Infinite data, finite memory)**.** *Let $M \geq 4$ and $d > 8\log(M)$. For any memory weight scheme $q : [N] \to \mathbb{R}$, when the embeddings e_x are independent random variables $e_x \sim \mathcal{N}(0, I)$, and the unembeddings are taken uniformly at random on the sphere,*

$$\mathbb{E}_{e,u}[\mathcal{E}(f_q)] \leq \inf_\gamma 2d^{-\gamma} + p\Big(\Big\{x \in [N] \,\Big|\, dq(x)^2 \leq 16 c_\gamma \Big(Q_\infty + \frac{8c_\gamma \|q\|_2^2}{d}\Big)\Big\}\Big), \tag{7}$$

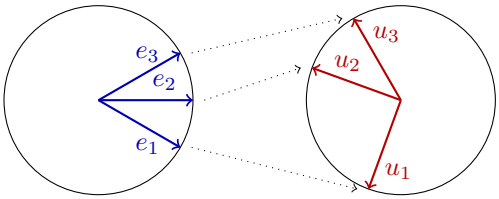

Figure 2: Error due to finite memory capacity: the stacking of associative memories in a matrix W may exhibit a pattern $W = \sum_x u_{f_*(x)} e_x^\top$ where three inputs mapped to three different outputs interact in such a way that $u_2^\top W e_1 = e_2^\top e_1 + u_2^\top u_3 e_3^\top e_1 \geq 1 + u_1^\top u_3 e_3^\top e_1 = u_1^\top W e_1$, so that $f_W(x=1) = 2 \neq 1 = f_*(x=1)$. In other terms, memory interference may lead to wrong prediction, illustrating the finite capacity of the model f_W (2) to store all data associations.

where $Q_\infty := \max_y \sum_{x; f_*(x)=y} q(x)^2$, $c_\gamma = \log(M) + \gamma \log(d)$, *and* $p(\mathcal{X}) = \sum_{x \in \mathcal{X}} p(x)$ *denotes the probability of* x *to belong to* $\mathcal{X} \subset [N]$. *In terms of lower bound,*

$$\mathbb{E}_{e,u}[\mathcal{E}(f_q)] \geq \frac{1}{20} p(\{x \in [N] \,|\, 3(d+1)q(x)^2 \leq Q_\infty\}). \tag{8}$$

Theorem 1 illustrates how the error made by a scheme q at the input x relates to the ratio between the signal $dq(x)$, provided by the associative memory $u_{f_*(x)} e_x^\top$, and the noise Q_∞, which corresponds to the signal provided by the most competitive class for $y \in [M]$. This is true up to a higher term in $\|q\|^2/d$, which corresponds to a class $y = f_*(x)$ competing against itself when the random embeddings $e_{x'}$ for x' such that $f_*(x') = y$ point in the opposite direction of e_x. When d is large and p is regular, $c_\gamma \|q\|_2^2/d$ will be dominated by Q_∞ and the cut-off of $q(x)^2/Q_\infty$ at $32 c_\gamma/d$ will behave similarly to a cut-off at $1/d$ up to logarithmic terms. Moreover, when q is chosen independently of $p(y|x)$,[2] one can expect $Q_\infty \approx p_* \|q\|^2$ where $p_* = \max_{y \in [M]} p(y)$. As a consequence, up to constants and logarithmic term, we get

$$\mathbb{E}[\mathcal{E}(f_q)] \approx p(\{x \in [N] \,|\, dq(x)^2 \leq p_* \|q\|^2\}). \tag{9}$$

3.2 Memory schemes

Let us now discuss several natural choices for q and compare their corresponding performance. The first naive choice consists in storing all the data seen at time T in memory. It reads

$$\hat{q}_0(x) = \mathbf{1}_{x \in \{X_t\}_{t \in [T]}}, \qquad q_0(x) = 1. \tag{10}$$

Here, $\hat{q}_0$ corresponds to the learned weighted scheme based on the T data, while q denotes an idealized limit when one has infinite data. In the idealized setting $Q_\infty(q_0) = N p_*$ where $p_* := \max_{y \in [M]} p(y)$. From Theorem 1, we deduce that $\mathcal{E}(f_{W_{q_0}})$ will follow two regimes: an overflow regime where $3(d+1) \leq N p_*$ and in essence the memory W_{q_0} is too full to recover any signal in it, and $\mathbb{E}_{e,u} \mathcal{E}(f_{W_{q_0}}) > 1/20$ (8); a infinite memory regime where $d \geq N$ and all associations $e_x u_{f^*(x)}^\top$ can be stored orthogonally to one another, and the error $\mathbb{E}_{e,u} \mathcal{E}(f_{W_{q_0}})$ quantifies the tiny probability that some random inputs embeddings appear to be too correlated.

Equipped with the knowledge that our associative memory model (2) has finite capacity, one may weight memories according to their frequencies, leading to the scheme, for $\rho \geq 0$

$$\hat{q}_\rho(x) = \Big(\frac{1}{T} \sum_{t \in [T]} \mathbf{1}_{x = X_t} \Big)^\rho, \qquad q_\rho(x) = p(x)^\rho. \tag{11}$$

A better option consists in explicitly limiting the storage of our model with a simple thresholding algorithm

$$\hat{q}_{\rho,[P]}(x) = \hat{p}(x)^\rho \mathbf{1}_{x \in \mathrm{top}_P((x_t)_{t \in [T]})}, \qquad q_{\rho,[P]}(x) = p(x)^\rho \mathbf{1}_{x \in [P]}, \tag{12}$$

where $\mathrm{top}_P((x_t))$ denotes the set made of the P most frequent inputs in the data (x_t).

[2] To be more precise, one should actually choose $q(x)$ to be class dependent so to cram in memory as many x as possible for each different class $y = f_*(x)$, ensuring that $y \mapsto \sum_{x; f_*(x)=y} q(x)^2$ is constant with respect to y. For simplicity, we will not discuss this behavior that does not change the big picture beyond our exposition.

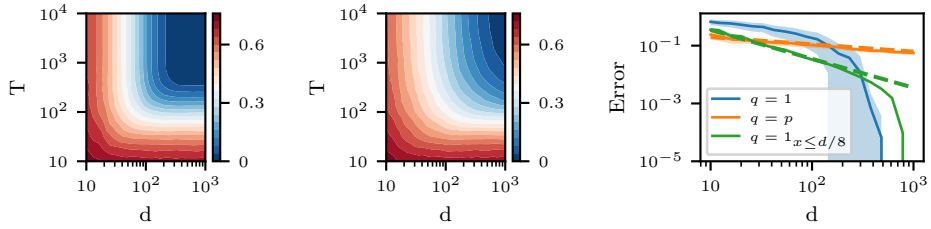

Figure 3: Generalization error (5) as a function of d and T for the model (4) averaged over 100 runs. The data follows a Zipf law with $\alpha = 0.5$, $N = 100$, $M = 5$ and $f_*(x) = x \bmod M$. Left: error for q_0 (10), either d is too small and there will be memory overflow leading to large error (red area), either it is big enough and with enough data, the error will be null (blue area). Middle: error for q_1 (11), for small d and big T, it avoid memory overflow allowing a smaller error then q_0; however for big d it does not allocated enough memory to rare association, leading to a bigger error. Those results can be interpreted mechanistically by looking at the corresponding memory matrices (see Figure 10). Right: Generalization error when $T = +\infty$, $N = 100$ and $\alpha = 2$: the scheme q_0 leads to a zero-one type of plot where if $d < N$ the error is high, and if $d > N$ the error decreases fast to zero (in blue); the scheme q_1 leads to an error decreasing in $d^{-(\alpha-1)/2\alpha} = d^{-1/4}$ as predicted by theory (in orange); the scheme $q_{0,P}$ (12) with $P = d/8$, decreases in $d^{-(\alpha-1)} = d^{-1}$ until reaching the tipping point when $d/8 > N$ (in green).

Proposition 2 (Without thresholding)**.** *Let p be an α-Zipf distribution* (1)*. For $\rho > 0$, the performance of $f_\rho := f_{q_\rho}$* (11) *is, up to poly-logarithm factors and constants that depends on both ρ and α,*

$$\mathbb{E}_{e,u}\mathcal{E}(f_\rho) \overset{(\log)}{\asymp} \left(\frac{d}{\varphi(N)}\right)^{-(\alpha-1)/2\rho\alpha}, \quad \textit{where} \quad \varphi(N) = \begin{cases} 1 & \textit{if } 2\rho\alpha > 1 \\ \log(N) & \textit{if } 2\rho\alpha = 1 \\ N^{1-2\rho\alpha} & \textit{if } 2\rho\alpha < 1 \end{cases}. \tag{13}$$

In particular, when $\rho = 1$, $\mathbb{E}_{e,u}\mathcal{E}(f_0)$ scales in $d^{-(\alpha-1)/2\alpha}$. In the limit where $\rho = 0$, $\mathbb{E}_{e,u}\mathcal{E}(f_0)$ can be understood as $(d/N)^{-\infty}$ which will go to zero if and only if d is bigger than N.

Proposition 3 (With thresholding)**.** *Assume that $p(x)$ follows a α-Zipf law* (1) *with $N = +\infty$. For $\rho \geq 0$, setting $P \simeq d^{1/(2\alpha\rho+1)}$, the error made by the memory scheme* (12) *scales as*

$$\mathbb{E}_{e,u}\mathcal{E}(f_\rho) \overset{(\log)}{\asymp} d^{-(\alpha-1)/(2\rho\alpha+1)}. \tag{14}$$

In particular, when $\rho = 0$ and $P \simeq d$, one gets a scaling in $d^{-\alpha+1}$, which is actually optimal. The fact that this maximum is reached for $P \simeq d$ is reminiscent of Hopfield networks (Hopfield, 1982) which can only store $d/\log(d)$ patterns with a d by d matrix. Similarly, our model stores at most d associations, which, when in presence of a Zipf law, leads to an error scaling in $d^{-(\alpha-1)}$.

Theorem 2 (Minimax performance)**.** *Assume that $p(x)$ follows a α-Zipf law* (1) *with $N = +\infty$. For any weighting scheme q, and $p_* \in (0, 1)$, there exists a conditional distribution $p(y|x)$ with $p_* = \max_y p(y)$ such that the error made for the distribution p is lower bounded by*

$$\mathbb{E}_{e,u}\mathcal{E}(f_q) \geq c_\alpha (d+1)^{-\alpha+1} \qquad \textit{where} \qquad c_\alpha = \frac{C_\alpha p_*^{\alpha-1}}{20(\alpha+1)\cdot 3^{\alpha-1}}.$$

Moreover, this performance is reached (up to logarithms factor) by the thresholding algorithm (12) *with $P \simeq d/\log(d)$ and $\rho = 0$.*

Finally, we prove that the scaling laws proved for d when $T = +\infty$ and for T when $d = +\infty$ appears jointly when both d and T are finite.

Proposition 4 (Finite data and finite memory)**.** *For the previous bound with respect to d, Proposition 2 and Proposition 3, considering finite data simply adds a term $T^{-1+1/\alpha}$ (up to constants and logarithmic terms), matching the optimal bound of Proposition 1. In particular,* (12) *with $\rho = 0$ and $P \simeq d/\log(d)$ reaches the optimal scaling in*

$$\mathbb{E}_{e,u,(x_t,y_t)_{t\in[T]}}\mathcal{E}(f_{\hat{q}}) \asymp T^{-1+1/\alpha} + d^{-\alpha+1}. \tag{15}$$

The optimal scaling (15) recovers the law of Hutter (2021) with respect to T, and the one of Michaud et al. (2023) with respect to d. This is intuitive, since Hutter (2021) assumes memorizing exactly

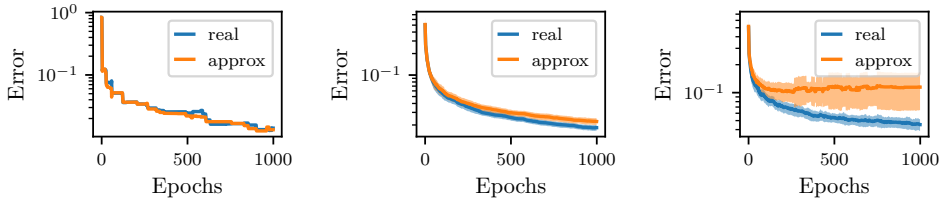

Figure 4: Comparison between the error found by optimizing W (2) with SGD on the cross-entropy loss, and its approximation with $q(x)$ (4) and the approximate update rule (20). We consider $N = 100$, $M = 5$, $f_*(x) = x \bmod M$, $\alpha = 2$, and batch size equals one. Left: One run with $d = N = 100$ with $\gamma = 10$. Middle: Average over 100 runs with $d = N = 100$ with $\gamma = 1$. Right: Average when $d = N/10 = 10$ with $\gamma = 1$, which implies that our approximation is not valid anymore. The same results can be obtained for bigger batch sizes as shown in Figure 13.

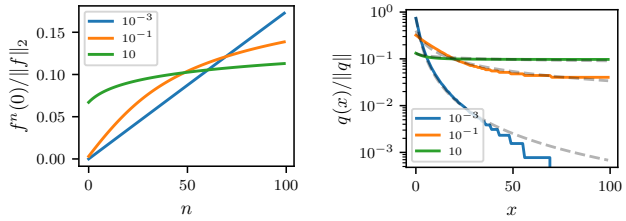

Figure 5: Theoretical approximation of the association scheme found with stochastic gradient descent with batch size equals one and fixed learning rates. Left: Plot of $f^n(0)$ as a function of n where f is the effect of one gradient update on $q(x)$ (20). Right: Plot of the resulting $q_\gamma(x)$ when $n_x \propto p(x) \propto (x+3)^{-\alpha}$ with $\alpha = 2$ and $n_N = 1$. In dashed, we represent q_ρ (11) for $\rho = 0.05$, $\rho = 0.35$ and $\rho = 1$. Those curves map well q_γ for $\gamma = 10$, $\gamma = 10^{-1}$ and $\gamma = 10^{-3}$ respectively.

all previously seen data, while each memory could be seen as specifying a "quantum of knowledge" as modeled in Michaud et al. (2023), with $d^{-\alpha+1}$ corresponding to the risk (5) of only storing the most frequent d tokens. However, associative memories can be understood at different level of granularity, and while one may argue that a transformer acts as a big associative memory machine and derives LLMs scaling laws approximations as corollaries, we prefer to understand a transformer as a combination of hidden associative memories as suggested by Sukhbaatar et al. (2019); Geva et al. (2021); Wu et al. (2022); Bietti et al. (2023) among others.

4 Optimization-based memorization

This section studies memory schemes privileged by optimization-based algorithms, digging into the training dynamics behind memorization. In terms of relevance, we argue that our model (2) is a proxy for the inner layers of a transformer that memorize patterns before matching them against new data at inference time. As such, we want to understand how different key elements in the training of a transformer influence storage in our memory model.

Gradient updates. We consider the cross entropy loss as a surrogate objective to minimize, and study the form of gradient updates on batches of data. Formally, the matrix $W \in \mathbb{R}^{d\times d}$ in (2) is optimized to minimize the loss

$$\mathcal{L}(W) = \mathbb{E}_{(X,Y)\sim p}[\ell(x, y; W)], \qquad \ell(x, y; W) = -u_y^\top W e_x + \log\Big(\sum_{z\in[M]} \exp(u_z^\top W e_x)\Big). \quad (16)$$

The gradient of this loss with respect to W takes the following form, as detailed in Appendix A.10:

$$\nabla_W \ell(x, y; W) = -(1 - p_W(y|x))(u_y - \varepsilon)e_x^\top, \quad \text{with} \quad \varepsilon = \sum_{z\in[M]} p_W(z|x, z \neq y)u_z. \quad (17)$$

where $p_W(y|x) \propto \exp(u_y^\top W e_x)$ are model predictions for the current W. For a batch of n data $B = [x_1, \cdots, x_n]$, a gradient update with step size γ_t updates W_t as

$$W_{t+1} = W_t - \gamma_t \sum_{x\in B} \nabla_W \ell(x, f_*(x); W_t). \quad (18)$$

Figure 6: Gradient descent dynamics from perspective of the matrix $(u_y^\top W_t e_x)_{y,x} \in \mathbb{R}^{M\times N}$ with $N = 10$, $M = 5$, $\alpha = 1.5$, $f_*(x) = x \bmod 5$, and $d = 5 < N$. A lighter color in the square (y, x) means a higher value of $u_y^\top W e_x$. The optimal W corresponds to two diagonal strips of yellow boxes (see Figure 15). The matrix W_t is updated with stochastic gradient descent with batch size equal to one. From time to time, stochastic gradient descent will hit an association that is not properly stored in memory yet (the red boxes). It will consequently update the weight matrix $W_t \to W_{t+1}$ (side by side pairs) to store it (18). Left pair: update with a big learning rate $\gamma = 10$, whose risk is to erase previous memories (the light colored boxes), similarly to q_0 (10). Right pair: update with a small learning rate $\gamma = 10^{-1}$, which will not store rare memory, similarly to q_ρ (11) with large ρ.

Approximation of the updates. When $p_W(z|x)$ does not change much for all $z \neq f_*(x)$, since u_z were sampled at random in $\mathcal{S}^d$, we expect ε (17) to concentrate around zero with $\|\varepsilon\|^2 \approx 1/M$, hence to be negligible in front of $u_{f_*(x)}$. As a consequence,

$$\nabla_W \ell(x, f_*(x); W) \approx -(1 - p_W(f_*(x)|x)) u_y e_x^\top. \tag{19}$$

This is notably the case for $W = 0$, random W, or if W only stores pairs $(x, f_*(x))$ with $d \gg N$. With the update model above (19), T steps of SGD with batch size one lead to an association scheme of the form (4) with (see Appendix A.11)

$$q_\gamma(x) \approx f^{Tp(x)}(0) = \underbrace{f \circ f \circ \cdots \circ f}_{Tp(x) \text{ times}}(0), \qquad \text{where} \qquad f : x \mapsto x + \frac{\gamma}{1 + M^{-1}\exp(x)}. \tag{20}$$

This equation tells us what form to expect for q for optimization schemes with different hyperparameters. This approximation is shown in Figure 5, and is validated empirically in Figure 4.

Step size effect. When $d > N$, the updates approximation (20) and the resulting q_γ show how a large learning rate γ is beneficial for our problem, in particular when using SGD with batch size one. Interestingly, the same behavior holds in the presence of limited capacity, i.e., $d < N$, although interferences between embeddings (Figure 2) break our approximation (19). In those settings, we resort to numerical simulation to study how optimization manages to rearrange memories. Figure 6 showcases two types of behaviors depending on the size of γ. *(i)* When the learning rate γ is large, associations will be stored easily in memory, but will tend to overwrite previous storage. *(ii)* When the learning rate γ is small, associations need to be seen often to build up in the matrix W (4) which will take more time, but will not erase memory. This provides another intuition explanation for why a bigger step size leads to better results on the left of Figure 7. The previous considerations also explain the usefulness of **scheduling** in our simple model, which we illustrate on Figure 11: using a large learning rate enables us to store associations while there is still memory space, while reducing it later in training avoids overwriting previous storage unless an association is highly frequent.

Batch size effect. Table 2 recalls how storing associations with $q = 1$ under the model (4) is better than storing them with $q = p$. As such, it suggests that, when processing a finite number of data T, smaller batch size is preferable. Intuitively, processing an input x in a batch will reweight it by its frequency $p(x)$, while processing it by itself will update W similarly to setting $q_\gamma(x) = 1$ if x has not been already seen. Indeed, in the large batch limit where $|B| \to +\infty$, one batch update corresponds to a population gradient update, which when $p_W \ll 1$ assimilates to $\nabla_W \mathcal{L}(W) \approx -\sum_x p(x) u_{f_*(x)} e_x^\top$. This contrasts with many small batch updates that rather lead to an association scheme akin to (4) with $q = 1$. In support of this line of reasoning, Figure 7 (middle) illustrates the benefits of splitting the descent with many steps, with a small batch size and large step size.

4.1 Practical considerations

In order to optimize our simple model the fastest, we have seen the usefulness of large step size and small batch size. However, for large transformers such design choices are impractical. First, large step sizes may lead to instability in realistic models (Gilmer et al., 2021). Second, in order to reduce training time and improve hardware efficiency, one should process large batches (Smith et al., 2018).

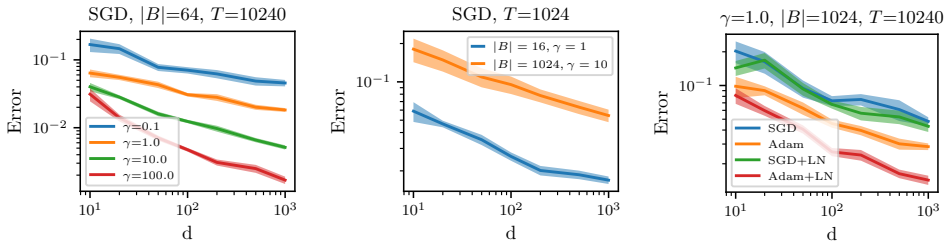

Figure 7: Effect of step size, batch size, layer-norm and Adam (with $\beta_1 = \beta_2 = 0$, which corresponds to SignGD). All the experiments are conducted with $N = 100$, $M = 5$, $\alpha = 2$, $f_*(x) = x \bmod M$, averaged over ten runs. We initialized parameters and rescale learning rates to ensure maximal feature updates, as explained in Appendix B.1. To avoid confounders, we scale γ on the middle plot for the variance of the gradient updates to be independent of the batch size.

Adam. We have seen before how the update of SGD with large batch can be approximated with

$$\gamma_t^{-1}(W_{t+1} - W_{t-1}) = \sum_{x \in B} (1 - p_W(f_*(x)|x)) u_{f_*(x)} e_x^\top \approx \sum_{x \in \mathbb{N}} |B|(1 - p_W(f_*(x)|x)) p(x) u_{f_*(x)} e_x^\top.$$

Those naive updates would lead to a model that resembles (4) with $q = p^\rho$ for $\rho \approx 1$ (11). In concordance with previous research on the matter (Zhang et al., 2020; Kunstner et al., 2023), we found Adam to be helpful in our setup as well, see Figure 7 (right). In first order approximation, Adam is approximated as signSGD (Balles & Hennig, 2018). Arguably, this introduces a normalization effect to the gradient, helping to reach the saturation phase of $n \mapsto f^n$ (20) shown on Figure 5, homogenizing the resulting matrix W to behave similarly to $q_1 = 1$, therefore optimizing memory capacity. Experiments to underpin this intuition are reported in Figures 15 and 16 in Appendix B.

Layer normalization. Minimizing the cross-entropy loss implies setting $p_W(y|x) = 1$, which will lead to W diverging to infinity and unstable loss gradients. In order to ensure numerical stability, it is natural to rescale the vector $We_x \in \mathbb{R}^d$, especially since what matters for the final prediction f_W is only its direction. This is precisely what layer-norm does, introducing the logit score

$$g_y^{\mathrm{LN}}(x) = \langle u_y, \frac{We_x}{\|We_x\|} \rangle, \qquad \text{instead of} \qquad g_y(x) = u_y^\top We_x.$$

This leads to an added projection on the gradients in (17), as detailed in Appendix A.12, denoting $\bar{W} = W/\|We_x\|$,

$$\nabla_W \ell^{\mathrm{LN}}(x, y; W) = \nabla_W \ell(x, y; \bar{W}) = \frac{1}{\|We_x\|} \left(I - (\bar{W}e_x)(\bar{W}e_x)^\top\right) \nabla_{\bar{W}} \ell(x, y; \bar{W}). \tag{21}$$

We recognize a projection that kills the signal that already aligns with We_x. We conjecture that this introduces a clipping effect on the corresponding $q(x)$, optimizing for memory storage, and explaining the good performance observed in the right of Figure 7.

4.2 The benefits of learning the embeddings

Taking a step back, Theorem 1 implies that our model with d^2 parameters, the matrix $W \in \mathbb{R}^{d \times d}$ (4), only memorize about $d/\log(d)$ associations $(e_x, u_y) \in (\mathbb{R}^d)^2$ of size $2d$. Intriguingly, Lemma 1 below states that an exponential number of quasi-orthogonal elements can be put in $\mathbb{R}^d$, an event that actually holds with high probability when embeddings are random, showcasing intrinsic limitations of our "linear" model (2).

Definition 1 (Quasi-orthogonality)**.** *The family $(u_z)_{z \in [P]}$ with $u_z \in \mathbb{R}^d$ is η-quasi orthogonal if*

$$\forall \{z, z'\} \subset [P], \qquad |\langle u_z, u_{z'} \rangle| \leq \eta, \qquad \textit{and} \qquad \|u_z\| = 1. \tag{22}$$

Lemma 1. *For any $d \in \mathbb{N}$ and $P \geq 3$, there exists an embedding $u : [P] \to \mathbb{R}^d$ such that the family $(u_z)_{z \in [P]}$ is $\eta = 2\sqrt{d^{-1}\log(P)}$-quasi orthogonal.*

As a consequence of Lemma 1, the following model

$$f_1(x) = \arg\max_y u_y^\top \sum_{x' \in [P]} u_{f_*(x')} \sigma(e_{x'}^\top e_x - \eta), \tag{23}$$

Figure 8: Experiments with learned embeddings when $\alpha = 2$, $N = 100$ and $M = 5$ with $y = f_*(x) = x \bmod M$ and $d = 2$. Left: level lines of the function $\mathbb{R}^2 \to [5]; u \mapsto \arg\max_{y \in [5]} u_y^\top u$ with u_y the learned unembedding. Middle: scatter plot of the learned input embeddings $e_x \in \mathbb{R}^2$ for $x \in [N]$ colored accordingly to $f_*(x)$ for e_x. It illustrates how the input embeddings match with the output ones, similarly to (24) and Proposition 5. Right: learned input embeddings obtained with $M = 10$, and allowing again a zero generalization error. Reaching a zero error with $d = 2$ greatly contrasts with the condition $d \geq N$ needed to get to a zero generalization error when the embeddings are random.

where $\sigma(x) = x_+$ is the ReLU function, can fit $P = \exp(\eta^2 d/4)$ elements in memory, leading to a scaling in $\mathcal{E}(f_1) \asymp \exp(-(\alpha - 1)\eta^2 d/4)$ when $p(x)$ follows a α-Zipf law.[3] Similarly, one could consider higher moments of $e_{x'}^\top e_x$ which has been the basis for modern Hopfield networks (Krotov & Hopfield, 2016; Ramsauer et al., 2021). However, implementing the model (23) requires to keep track of each of the P vectors $e_x \in \mathbb{R}^d$, leading to Pd parameters, in order to only store P associations of size d, needing compute that scales with Pd at inference time, rather than just d^2,

We also note that when embeddings are learned, it is actually possible to store as many memories as desired, which can be seen from the fact that

$$W = I, \forall\, y \in [M]\, u_y \in \mathcal{S}^d, e_x = u_{f_*(x)} \qquad \Rightarrow \qquad f_*(x) = \arg\max_y u_y^\top W e_x, \tag{24}$$

In particular, Figure 8 illustrates the solution found when $d = 2$ by optimization-based algorithms in order to get a zero generalization error on the task of Figure 3 where $M = 5$. Optimizing token embeddings is probably an important element to increase memorization capacity in transformers, although enforcing $e_x = u_{f_*(x)}$ is unrealistic when embeddings are shared over different heads, and the input/output relationships to be learned differ across heads.

5 Conclusion

This work considers a simple model to study memorization in transformers. Here, memorization is seen as a valuable behavior, the network memorizing useful patterns and association rules. We derive precise scaling laws with respect to both the number of data, and the model size, which plays the role of a model capacity. We quantify the effect of different memorization schemes, illustrating the benefits of uniformly weighted outer products. We leverage these theoretical results to study how different optimization algorithms commonly used for transformers may lead to more efficient memorization. In particular, we showcase the efficacy of small batches and large learning rates, and, under the design constraints resulting from efficient hardware utilization and training stability, the usefulness of Adam and layer normalization.

While our study focuses on simple memorization schemes, it opens up many possible new directions. This includes extending our study to richer models that are closer to transformers, where embeddings, attention and feed-forward layers are trained. This could allow models of scaling laws that capture interactions between tokens, as well as hierarchical behaviors that require multiple layers. We would equally like to leverage our framework for assessing memorization and generalization through clear metrics, and eventually automatically adapt the learning rates as a function of the "free" memory capacity left in a layer.

References

Shun-Ichi Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, 1972.

[3] This result follows directly from two facts. When input embeddings are chosen at random, the probability that they are not η-quasi orthogonal is bounded by $P^2 \exp(-d\eta^2/2)$. When input embeddings are η-quasi orthogonal, $f_1(x) = f_*(x)$ for any $x \in [P]$.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *ICML*, 2018.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. In *NeurIPS*, 2023.

Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley, 1991.

Lukasz Debowski. A simplistic model of neural scaling laws: Multiperiodic santa fe processes. *arXiv preprint arXiv:2302.09049*, 2023.

Ian Dinwoodie. Mesures dominantes et théorème de sanov. *Annales de l'Institut Henri Poincare*, 1992.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. Technical report, Anthropic, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. Technical report, Anthropic, 2022.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *STOC*, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *NeurIPS*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *EMNLP*, 2021.

Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Edward Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2021.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 1963.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *NeurIPS*, 2022.

John Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 1982.

Marcus Hutter. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

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.

Teuvo Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, 1972.

Dmitry Krotov and John Hopfield. Dense associative memory for pattern recognition. In *NeurIPS*, 2016.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. In *ICLR*, 2023.

William Little. The existence of persistent states in the brain. *Mathematical Biosciences*, 1974.

Christopher Longuet-Higgins, David. Willshaw, and Peter Buneman. Theories of associative recall. *Quarterly Reviews of Biophysics*, 1970.

Alexander Maloney, Daniel Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *NeurIPS*, 2022.

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *arXiv preprint arXiv:2303.13506*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Steven Piantadosi. Zipfs word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin and Review*, 2014.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlovi, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *ICLR*, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *ICML*, 2021.

Samuel Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *ICLR*, 2018.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artifical Intelligence*, 1990.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In *NeurIPS*, 2022.

Karl Steinbuch. Die Lernmatrix. *Kybernetik*, 1961.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

Juan Valle-Lisboa, Andrés Pomi, and Eduardo Mizraji. Multiplicative processing in the modeling of cognitive activities in large neural networks. *Biophysical Reviews*, 2023.

David Willshaw, Peter Buneman, and Christopher Longuet-Higgins. Non-holographic associative memory. *Nature*, 1969.

Yuhuai Wu, Markus Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *ICLR*, 2022.

Greg Yang and Etai Littwin. Tensor programs ivb: Adaptive optimization in the infinite-width limit. *arXiv preprint arXiv:2308.01814*, 2023.

Greg Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. In *NeurIPS*, 2021.

Jingzhao Zhang, Sairaneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In *NeurIPS*, 2020.