Memory Mosaics at scale

Jianyu Zhang
New York University, New York
FAIR, Meta Inc., New York

Léon Bottou
FAIR, Meta Inc., New York
New York University, New York

Abstract

Memory Mosaics [Zhang et al., 2025], networks of associative memories, have demonstrated appealing compositional and in-context learning capabilities on medium-scale networks (GPT-2 scale) and synthetic small datasets. This work shows that these favorable properties remain when we scale memory mosaics to large language model sizes (llama-8B scale) and real-world datasets.

To this end, we scale memory mosaics to 10B size, we train them on one trillion tokens, we introduce a couple architectural modifications ("*memory mosaics v2*"), we assess their capabilities across three evaluation dimensions: training-knowledge storage, new-knowledge storage, and in-context learning.

Throughout the evaluation, memory mosaics v2 match transformers on the learning of training knowledge (first dimension) and significantly outperforms transformers on carrying out new tasks at inference time (second and third dimensions). These improvements cannot be easily replicated by simply increasing the training data for transformers. A memory mosaics v2 trained on one trillion tokens still perform better on these tasks than a transformer trained on eight trillion tokens.

1 Introduction

In Machine Learning, compositional capabilities and in-context/out-of-distribution learning capabilities have been continuously pursued but remain challenging. Early attempts to achieve these goals include pursuing disentanglement via various statistical "independence" [Comon, 1994, Roth et al., 2022], pursuing out-of-distribution/learning from the perspective of optimization on multiple environments [Finn et al., 2017, Arjovsky et al., 2019, Bengio et al., 2019]. In contrast, transformer-based models demonstrate certain compositional capabilities and early in-context learning abilities. However, we still lack a clear understanding of how current transformers achieve these capabilities, and why earlier models were unable to.

Memory mosaics [Zhang et al., 2025], networks of simple key-value associative memories (without position encoding), offer a comparatively transparent way to understand how composition or disentanglement occur. Trained and evaluated on medium-scale networks and synthetic datasets, memory mosaics reveal promising superior in-context learning abilities. Therefore, we ask *"To peruse a strong and general new task learning capability, how can we scale memory mosaics to large networks and real-world datasets?"*

The contribution of this work: 1) We successfully scale up memory mosaics to llama-8B scale, using one trillion real-world training tokens. The resulting network is named as **Memory Mosaics v2**.[1] [2] Compared to memory mosaics, memory mosaics v2 made three architectural modifications, including an adaptive bandwidth of associative memory, a gated time-variant key feature extractor, and a 3-level memory design. 2) We propose three evaluation dimensions to comprehensively assess model ability (from i.i.d. to o.o.d. scenarios). 3) Our memory mosaics v2 demonstrate superior new-task learning capabilities (with fewer examples and less priori knowledge from human designers).

---

[1] For clarity, "Memory Mosaics" refers to the version in Zhang et al. [2025], while "Memory Mosaics v2" refers to this scaled-up version.

[2] `https://github.com/facebookresearch/MemoryMosaics`

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

This paper is organized as follows. Section 2 introduces background knowledge on associative memories. Section 3 presents the architecture of memory mosaics v2. Then section 4 describes the training process, section 5 evaluates memory mosaics v2 and transformers across three dimensions – training (persistent) knowledge storage, new knowledge storage, and in-context learning. Section 6 discusses the failure of replicating memory mosaics v2 by simply increasing the training data (×8 more data) for transformers. Section 7 studies the advantage of memory mosaics v2 in fine-tuning. Finally, section 8 provides discussion and further directions.

2 Background on Associative Memory

General speaking, memory mosaics architecture [Zhang et al., 2025] replaces attention blocks in transformers [Vaswani et al., 2017] with associative memories. This section provides the background on associative memories, highlights the connection and differences between "associative memory" in memory mosaics and "attention" in transformers.

Associative Memory Associative memories have a long history in both psychology and computer science, referring to relationships between unrelated items. In this work, we follow the definition from Zhang et al. [2025], according to which an associative memory is a device that can store key-value pairs $\{(k_1, v_1) \dots (k_n, v_n)\}$ and retrieve values given a corresponding key:[3]

$$k \mapsto f(k;\ \{(k_1, v_1) \dots (k_n, v_n)\}) \tag{1}$$

The key-value pairs are stored in a set, and thus can be assumed to be permutation invariant. This exchangeability property suggests that we can view an associative memory as a device that estimates a conditional probability distribution $P(V|K)$ on the basis of the sample $(k_1, v_1) \dots (k_n, v_n)$ of key-value pairs. The retrieval function is then a conditional expectation over this estimated distribution:

$$f\big(k;\ \{(k_1, v_1) \dots (k_n, v_n)\}\big) \ = \ \mathbb{E}(V \mid K = k)\,. \tag{2}$$

This conditional expectation can be estimated by kernel regression, e.g. Gaussian kernel regression:[4]

$$f\big(k;\ \{(k_1, v_1) \dots (k_n, v_n)\}\big) \ = \ \sum_{i=1}^{n} \frac{e^{-\beta\|k-k_i\|^2}}{\sum_{i=1}^{n} e^{-\beta\|k-k_i\|^2}}\ v_i\ , \tag{3}$$

where β controls the bandwidth of Guassian kernel.

Connection between associative memory and attention Associative memory in Equation 3 is closely connected to attention [Bahdanau et al., 2015] when all key vectors k_i share the same squared norm. That is, expression (3) becomes: [5]

$$f\big(k;\ \{(k_1, v_1) \dots (k_n, v_n)\}\big) \ = \ \sum_{i=1}^{n} \frac{e^{\,\beta\, k^\top k_i}}{\sum_{j=1}^{n} e^{\,\beta\, k^\top k_j}}\ v_i\ . \tag{4}$$

Moreover, the size of associative memory (i.e., the number of key-value examples) is analogous to the sequence length in attention.

Differences between associative memory and attention The associative memory viewpoint is conceptually simple and transparent. This simplicity contributes several key differences in associative memory (compared with attention), including: 1) L_2 normalized key vectors with an explicit bandwidth parameter β, 2) a symmetric kernel with the same formula for keys as queries, and 3) the absence of explicit position encoding. These differences further contribute to the superior compositional capabilities and in-context learning capabilities in memory mosaics.

3 Memory Mosaics v2

Transformers show an interesting induction head mechanism [Olsson et al., 2022], that is, predict b after sequence $[\dots, a, b, \dots, a]$. This mechanism contributes to their in-context learning ability. According to Bietti et al. [2023], position encoding and asymmetric query-key extractors (e.g.

---

[3] both keys and values shall be vectors in $\mathbb{R}^d$.

[4] This Gaussian kernel smoothing not only converges to the true conditional expectation $\mathbb{E}(K|V)$ when $n \to \infty$ and $\beta = \sqrt{n}$ [Nadaraya, 1964, Watson, 1964], but also makes it easy to compute gradients.

[5] For the easy of reading, we reuse the "attention score" notion to $\frac{e^{\,\beta\, k^\top k_i}}{\sum_{j=1}^{n} e^{\,\beta\, k^\top k_j}}$ in associative memories.

$q_T = W_q x_T$, $k_T = W_k x_T$) are essential for transformers to achieve this induction head mechanism with at least two layers of attention.

Inspired by studies of induction head mechanism, Memory Mosaics [Zhang et al., 2025] construct associative memories using keys to represent the recent past and values to represent the near future (Figure 2 left):

$$k_T = \varphi_\theta(x_T, x_{T-1}, \dots), \quad v_T = \psi_\theta(x_{T+1}, x_T, \dots) \tag{5}$$

This simple designer allows memory mosaics to get ride of explicit position encoding, use the same key as query, perform induction head with only one layer. The resulting memory mosaics also reveal appealing in-context learning capabilities on small synthetic datasets.

Based on memory mosaics, this section introduces **Memory Mosaics v2**, aiming at a stronger and more general in-context learning ability on broader real-world tasks (without loss of performance on other common benchmarks). Compared to memory mosaics, memory mosaics v2 incorporates three architecture modifications, including an adaptive bandwidth in associative memory, a gated time-variant key feature extractor, and a 3-level memory design.

3.1 Adaptive bandwidth in Gaussian kernel smoothing

Memory mosaics use one fixed bandwidth parameter β for different sizes n of associative memory (Equation 1). It is well known that bandwidth controls the bias-variance trade-off [Hastie et al., 2009] of kernel regression (memory-based) methods. That is, for a given distribution, the optimal bandwidth depends on the number of examples (key-value pairs in associative memory). Inspired by the asymptotic Mean Integrated Squared Error kernel bandwidth estimation approach where $1/\sqrt{\beta} \propto n^{-1/(p+4)}$ [García-Portugués, 2024], memory mosaics v2 schedule β in Equation 4 as:

$$\beta = \beta_1 n^\alpha + \beta_0 \,, \tag{6}$$

where $\beta_0 \geq 0$, $\beta_1 > 0$, $1 > \alpha > 0$ are learnable parameters (Check Appendix Table 6 for reparameterization and initialization details). I.e., the more key-value pairs (examples), the smaller bandwidth $1/\sqrt{\beta}$.

3.2 Gated time-variant key feature extractor

Memory mosaics employs a simple time-invariant leaky averaging to extract key features:

$$k_T = \mathrm{Norm}\big(\bar{k}_T\big) \quad \text{with} \quad \bar{k}_T = \tilde{k}_T + \lambda\, \bar{k}_{T-1} \qquad \tilde{k}_T = W_\varphi\, x_T \tag{7}$$

The averaging weights in Equation 7 are fixed and independent of the semantic input x. As a result, semantically similar cases, such as "tom-and-jerry" and "tom- - -and- - -jerry", may receive different key features. Inspired by recurrent-style networks [Peng et al., 2023, Gu and Dao, 2023, Beck et al., 2025], memory mosaics v2 utilize the following gated time-variant key feature extractor: [6]

$$k_T = \mathrm{Norm}\big(\bar{k}_T\big) \quad \text{with} \quad \begin{cases} \bar{k}_T = g_T \tilde{k}_T + \lambda_T \bar{k}_{T-1} & \tilde{k}_T = W_\varphi\, x_T \\ g_t = e^{W_g x_T} \in \mathbb{R}\,, \ \lambda_T = e^{-|W_\lambda x_T|} \in \mathbb{R} & \end{cases} , \tag{8}$$

where $W_\varphi, W_g, W_\lambda$ are learnable parameters, the averaging weights $\lambda_T \in \mathbb{R}$ and the exponential gate $g_T \in \mathbb{R}$ semantically depend on input x_T. See Appendix Figure 9 for graphical illustrations.

For key feature extractor, memory mosaics v2 reuses the same convolutional key extractor as in memory mosaics:

$$v_T = \alpha_\psi \mathrm{Norm}\big(\bar{v}_T\big) \quad \text{with} \quad \bar{v}_T = \gamma\, \tilde{v}_T + (1-\gamma)\, \tilde{v}_{T+1} \qquad \tilde{v}_T = W_\psi\, x_T \,, \tag{9}$$

where $\gamma, \alpha_\psi \in \mathbb{R}$ and W_ψ are learnable parameters.

3.3 3-level memory

Transformer architecture [Vaswani et al., 2017] consists of attention blocks and feedforward neural network blocks. The former handles local contextual information from an input sequence, while the latter stores global persistent information shared by different training sequences. Memory mosaics [Zhang et al., 2025] simplify the attention and the feedforward network in transformer as contextual associative memory and persistent memory, respectively. This simplification reduces the dependence between the "attention score" and the token position, as shown in Figure 1. Compared with transform-

---

[6] It worth noting that this work is neither a linearization of attention nor attention efficiency. The recurrent feature extractor in Eq. 7 is used to create keys, while associative memory in Eq. 1 still stores all key-value pairs.

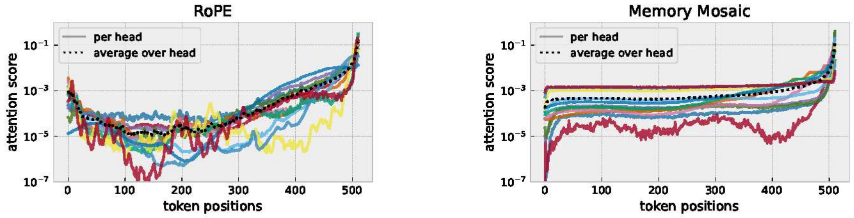

Figure 1: Average attention scores of the last token attending previous tokens. **Left**: Transformer with RoPE position encoding. **Right**: Memory Mosaics [Zhang et al., 2025]. The (averaged) attention scores in transformer heavily depends on token positions (curly curves), while the attention scores in memory mosaics at far tokens (e.g. position 0 to 450) are almost invariant to positions (flat curves).

ers (Figure 1 left), the attention scores in memory mosaics (Figure 1 right) exhibit a structured pattern. That is, attention scores on near-tokens (positions) heavily depend on positions, while attention scores on far-tokens are almost invariant to token positions. Inspired by this experimental discovery, memory mosaics v2 replace each contextual associative memory in memory mosaics with two associative memories, *short-term memory* and *long-term memory*, using distinct parameters (as in Figure 2).

Short-term memory The short-term memory at position t only stores key-value pairs of near-tokens, ranging from $t-h+1$ to $t-1$, implementing Eq. (4) as:

$$f\big(k;\ \{(k_{t-h+1}, v_{t-h+1}) \dots (k_{t-1}, v_{t-1})\}\big) \;=\; \sum_{i=t-h+1}^{t-1} \frac{e^{\beta\, k^\top k_i}}{\sum_{j=t-h+1}^{t-1} e^{\beta\, k^\top k_j}}\, v_i\,. \tag{10}$$

Long-term memory In contrast, the long-term memory skips near tokens and only stores key-value pairs before position $t-m$, implementing Eq. (4) as $f(k;\ \{(k_1, v_1) \dots (k_{t-m}, v_{t-m})\})$.[7] By setting $m < h$, memory mosaics v2 create an overlap between long-term and short-term memory, resulting in a soft boundary between these two memories. Eventually, the outputs of many long-term memories and short-term memories are concatenated together, following by a linear projection W_o.

Persistent Memory Memory mosaics v2 implements *persistent memory* using dense two-layer neural networks with SwiGLU activation [Shazeer, 2020] due to computational efficiency concerns.[8]

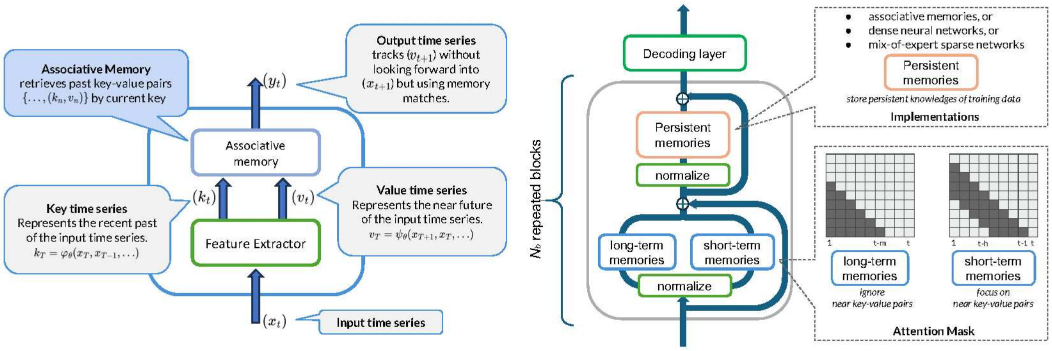

Figure 2: Left: Memory Unit. **Right:** Memory Mosaics v2 architecture.

4 Training

We train two memory mosaics v2 of difference sizes (small/large). Memory mosaics v2 small (llama-1.5B scale) contains 24 layers, 2048 hidden dimensions, and 16 heads, trained on 200 billion tokens of a diverse datamix. Memory mosaics v2 large (llama-8B scale) increases the number of layers to 32, hidden dimensions to 4096, and the number of heads to 32, trained on 1 trillion tokens of the same datamix. Both models are trained on 4,096 context length, followed by a fine-tuning process on 32,768 context length. Other training details are provided in Appendix C.

---

[7] Note that k, v, and β in long-term and short-term memory are constructed with distinct parameters.

[8] A two-layers feed-forward network and a key-value associative memory are interchangeable as shown in Sukhbaatar et al. [2019].

Stochastic long-term memory size During training, memory mosaics v2 samples the long-term memory delay step m from $[64, 256]$, sets the short-term memory window size $h = 256$. At inference, m is set to 64. This stochastic long-term memory training setup encourages the allocation of position-invariant signals to long-term memory and position-dependent signals to short-term memory (as shown in Figure 1). The experimental results in Appendix G Table 12 show that this training setup enhances context-length extrapolation ability by more than 15%.

Baseline We train two baseline transformers (small/large) with the same configurations as their memory mosaics v2 counterparts. Unless otherwise specified, in this work, transformer models use llama architecture [Grattafiori et al., 2024] with multi-head attention.

5 Three evaluation dimensions

The evaluation design provides a means to assess the specific properties of a system. Memory mosaics v2 aims at the ability to learn new tasks with fewer examples and less task-specific priori knowledge [Zhang, 2025]. Thus, to fully assess this capability, this section adopts three evaluation dimensions.

- **Persistent-knowledge storage and retrieval**, the ability of persistent-memory to store and retrieve knowledge of training dataset. This capability prepares knowledge that could be reused in other tasks during inference. We use common language benchmarks to access this aspect.
- **New-knowledge storage and retrieval**, the ability to store and retrieve new information of test dataset. It is a prerequisite for "learning" new tasks via memory-based methods. We employ "multi-unrelated-documents storing and question-answering" tasks to evaluate this aspect.
- **In-context Learning**, directly evaluates the ability to learn new tasks with fewer examples and less task-specific priori knowledge. We use multiclass classification to assess this aspect.

5.1 Persistent-knowledge storage and retrieval

Table 1 evaluates both memory mosaics v2 and baseline transformers on 19 commonly used language benchmarks, showing that they perform closely on these benchmarks.This is expected since both models share the same persistent memory architecture.

Table 1: Memory mosaics v2 and transformers performance on 19 common language benchmarks.

model	context length	obqa	arc easy	wino-grande	arc challenge	piqa	boolq	hell-aswag	nq	siqa	tqa	gsm8k	mmlu alt	human eval+	squad	bbh	math	mbpp	race middle	race high	avg
transformer small	32k	35.2	61.0	60.1	31.4	73.6	63.0	59.3	11.7	44.5	26.7	3.0	35.2	32.4	54.7	26.0	1.2	9.2	52.2	37.4	37.8
memory mosaics v2 small	32k	35.0	60.0	58.4	32.9	73.3	62.7	58.0	11.8	46.6	29.3	3.1	34.7	30.8	59.3	27.3	1.1	9.4	49.2	38.4	38.0
transformer large	32k	45.8	77.3	72.3	52.6	80.8	72.6	79.2	31.9	49.3	61.5	32.4	49.0	38.3	76.3	45.6	8.7	9.8	62.6	45.6	52.2
memory mosaics v2 large	32k	45.4	78.0	71.2	51.8	80.4	73.1	78.6	30.9	48.6	62.0	27.4	48.2	43.0	78.2	47.8	8.8	9.6	61.6	46.5	52.2

How do we know whether these benchmarks access persistent-knowledge ability rather than new-knowledge ability? To answer this question, we re-evaluate these benchmarks on memory mosaics v2 but with *long-term memory* being removed after training. The underlying reason is that if a task solely relies on the information stored in persistent memory and retrieved by short-term memory, removing long-term memory should not significantly affect performance.

Table 2 shows that removing long-term memory after training does not degrade the performance of 13 common benchmarks. This suggests that these 13 tasks are almost exclusively based on information stored in persistent memory and retrieved by short-term memory. In contrast, Appendix Table 9 indicates that the other 6 benchmarks perform poorly when long-term memory is removed.

Based on these findings, we use the 13 tasks to evaluate persistent knowledge storage and retrieval capability. The results (Table 1) show that memory mosaics v2 and transformers perform similarly in this evaluation dimension, suggesting that both models are capable of effectively storing and retrieving persistent knowledge.

Table 2: Memory mosaics v2 performance on 13 common language benchmarks. Removing the "long-term memory" after training barely hurt the performance (56.6% vs 56.8%). Flops/token is estimated at context length 256 via tha approach of Casson [2023].

	params	flops/token	obqa	arc easy	wino-grande	arc challenge	piqa	boolq	hell-aswag	nq	siqa	tqa	gsm8k	mmlu alt	human eval+	avg
Transformer large	8.8B	16.7B	45.8	77.3	72.3	52.6	80.8	72.6	79.2	31.9	49.3	61.5	32.4	49.0	38.3	57.1
memory mosaics v2 large	9.9B	18.9B	45.4	78.0	71.2	51.8	80.4	73.1	78.6	30.9	48.6	62.0	27.4	48.2	43.0	56.8
memory mosaics v2 large without long-term memory	8.3B	15.6B	45.4	77.9	71.2	51.8	80.4	73.1	78.6	30.8	48.6	62.1	26.7	46.8	42.2	56.6

Computation and # parameters concerns Table 2 summarizes the size of parameters and computation required for transformers and memory mosaics v2. Interestingly, removing long-term memory from memory mosaics v2 after training achieves a comparable transformer performance on the 13 persistent-knowledge benchmarks, while using fewer parameters and computations.

5.2 New-knowledge storage and retrieval

The new-knowledge storage and retrieval ability is a prerequisite for learning new tasks via memory-based methods (e.g., Gaussian kernel regression), because the data of new tasks must be adequately "stored" before learning (Note that memory-based methods are lazy methods). To illustrate this point, consider a poor goldfish with 7-second memory – how can it possibly learn a 90-minute movie? Similarly, a model with limited new-knowledge storage ability will struggle to learn information that exceeds its storage (memory) capacity.

Task description To assess this ability, we employ two "multi-unrelated-documents question-answering" tasks from the RULER benchmark [Hsieh et al., 2024]. These tasks involve multiple concatenated realistic articles followed by a question related to one of these articles, requiring the model to find the correct answer based on the correct article.[9] A prompt example is:

> Answer the question based on the given documents. The following are given documents. Document 1: [...] Document2: [...] [...] Document 20: [...] Question: What religion were the Normans? Answer:

These tasks are notably more challenging than typical 'needle-in-a-haystack' benchmarks [Kamradt, 2023], owing to their high information entropy. The typical 'needle-in-a-haystack' task is too easy, resulting in many models achieving near-perfect performance. See Table 13 in Appendix for details.

Main results Table 3 compares memory mosaics v2 and transformers, pretrained on a 4k context length, on these question-answer tasks. Memory mosaics v2 outperforms transformers on 4k task-length by 1.4%$\sim$5.6%. Similarly, Table 4 presents the same comparison, but with both models fine-tuned at a 32k context length. As task lengths increase to 32k, the "multi-unrelated-documents question-answering" tasks become more challenging. At this increased difficulty level, memory mosaics v2 significantly outperforms transformers by 12.3% to 14.8%.

Table 3: Comparison of memory mosaics v2 and transformer, trained on 4k context length, on RULER question-answer tasks. Memory mosaics v2 not only outperforms transformer on 4k task-length, but also successfully extrapolate the context length $\times 4 \sim \times 8$ times without any fine-tuning.

model	context length	task-length 4k	8k	16k	32k
transformer small	4k	39.4	×	×	×
memory mosaics v2 small	4k	45.0	35.0	34.1	31.7
transformer large	4k	57.7	×	×	×
memory mosaics v2 large	4k	59.3	48.8	46.4	26.5

Table 4: Comparison of memory mosaics v2 and transformer, trained on 4k and fine-tuned on 32k context length, on RULER question-answer tasks. Memory mosaics v2 outperforms transformer by 12.3%$\sim$14.8%.

model	context length	4k	8k	16k	task-length 32k	64k
transformer small	32k	37.0	29.3	29.0	22.1	×
memory mosaics v2 small	32k	44.3	39.3	39.4	36.9	25.3
transformer large	32k	51.2	48.8	44.7	41.1	×
memory mosaics v2 large	32k	58.9	55.5	54.9	53.4	46.4

The failures of many potential baselines Many memory compression algorithms, such as RNNs, xLSTM [Beck et al., 2025], rwkv [Peng et al., 2023], and state-space models [Gu and Dao, 2023], fail on this task by construction because they cannot store all articles before reading the question. Similarly, local-window memory approaches, such as Alibi position encoding Press et al. [2021] and sliding-window attention Beltagy et al. [2020], also struggle for the same reason.[10] This incompetent

---

[9] Similarly to the process used in section 5.1 for verifying persistent-knowledge storage and retrieval tasks, appendix Table 10 compares memory mosaics v2 with and without long-term memory on these question-answering tasks, confirming the necessity of "long-term memory" for these tasks.

[10] One might argue to play around this shortage by reading the question before the multiple articles. However, this process involves task-specific priori knowledge from human designers. In the end, instead of proving the

of memory compression algorithms has also been experimentally demonstrated by Hsieh et al. [2024] and Li et al. [2024]. Also, see Appendix G for these experimental evidences.

Extrapolating context length (without fine-tuning) Context length extrapolation (without fine-tuning) not only is computationally appealing, but also reveals the model's consistency in handling context. Unfortunately, transformers (with ROPE position encoding) struggle to extrapolate context length, as shown in Table 3.[11] In contrast, memory mosaics v2, trained on 4k context length, not only outperform transformers on 4k length, but also perform well after extrapolating context length $\times 4 \sim \times 8$ times without any fine-tuning or adaptation.

5.3 In-context learning

Having demonstrated the new-knowledge storage and retrieval ability of memory mosaics v2, this section takes a step further to evaluate its capacity to learn new tasks or distributions at inference time. This ability is also commonly referred to as in-context learning.

Tasks description To assess the in-context learning ability, we employ classic multiclass classification problems,[12] adopted from Li et al. [2024]. The classification tasks include:

- **Banking77** [Casanueva et al., 2020] is a banking-intent classification task with 77 target categories. Each example has an average length of 24 tokens.
- **Tacred** [Zhang et al., 2017] is a relation classification task of two objects in a sentence, extracted from newswire or webtext, with 41 target categories. Each example has an average length of 77 tokens.
- **Goemotion** [Demszky et al., 2020] is an emotion classification task of Reddit comments with 28 target categories. Each example has an average length of 26 tokens.

To solely evaluate the ability to learn new tasks (reduce the influence of training knowledge), we create an anonymous version with anonymous target labels (e.g. "class 1", "class 2") for each classification task. The original classification setup with semantic labels (e.g. "happy", "angry") is referred to as semantic version.

In this section, we adopt a few-shot learning setup where each "shot" consists of one (x, y) example from each possible target label category. By collecting multiple shots, we create an n-shot classification task. To encode these (x, y) examples for memory mosaics v2 and transformers, we serialize the (x, y) pairs into a sequence followed by a test query x_{test}.[13] A prompt example is:

Given a customer service query, please predict the intent of the query. [...] The examples are as follows: query: x_{shot1}, instant: y_{shot1}, [...], query: x_{shot2}, instant: y_{shot2}, [...], query: x_{test}, instant:

Main Results Figure 3 compares the performance of memory mosaics v2 and transformers in three classification tasks with semantic target labels. The horizontal axis represents the number of shots, while the vertical axis represents the classification accuracy on x_{test}. We can observe two phenomena: **1)** memory mosaics v2 consistently improve classification performance as it sees more demonstration shots (blue curves). In contrast, transformers struggle to maintain their performance and exhibit counterintuitively degraded performance as more demonstrations are provided (red curves). **2)** Memory mosaics v2 significantly outperform transformers by more than 10%. Appendix H provides

---

machine is intelligent, it often proves that human designers are intelligent. Please recall that *a child does not prepare all questions before going to school*.

[11] The comparison ignores many memory compression and local window approaches [Press et al., 2021, Beltagy et al., 2020], because they fail on this evaluation by construction.

[12] We choose classic classification problems over other fancy benchmarks for two reasons. Firstly, the mechanisms underlying classification are well-studied, allowing us to confidently attribute good or poor performance to the system's properties. Secondly, classification tasks can be designed to be arbitrarily different from the training set by changing the classification boundary, making it easier to measure the ability to learn new distributions. In contrast, as of this writing, many fancy benchmarks may not offer the same level of control and fine-grained analysis.

[13] Transformers are known to be sensitive to the prompt strategies [Gupta et al., 2024, Mirzadeh et al., 2024], such as the delimiter before x and y, shuffling/not-shuffling the (x, y) examples within each shot. To reduce the influence of prompt strategies, we evaluate each classification task with different delimiters ("[space]" and "$\backslash n$"), shuffled/non-shuffled (x, y) examples. Then choose the best prompt strategy for each n-shot classification task. Check appendix I for prompt examples.

a similar comparison on a smaller model size (~1.5B), with an even larger margin. Appendix E further summarizes the comparison under matched model size or computation (FLOPs).

Figure 4 presents a similar comparison as Figure 3, but on anonymous target labels. Again, memory mosaics v2 significantly outperforms transformers on all classification tasks.

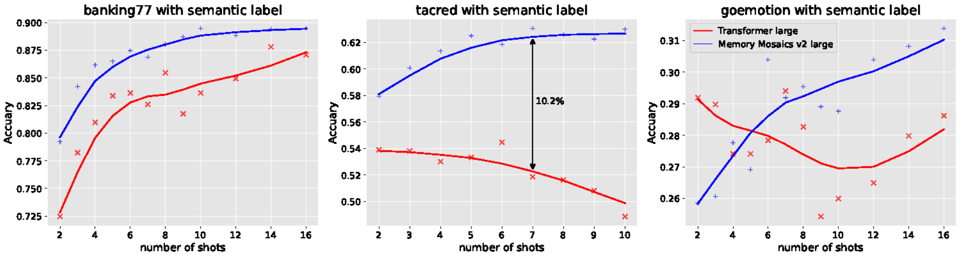

Figure 3: Semantic label in-context learning comparison between memory mosaics v2 and transformer. Memory mosaics v2 significantly outperform transformers on in-context learning with a large margin (more than 10%). Meanwhile, memory mosaics v2 benefits from more demonstration shots (x-axis), unlike transformers.

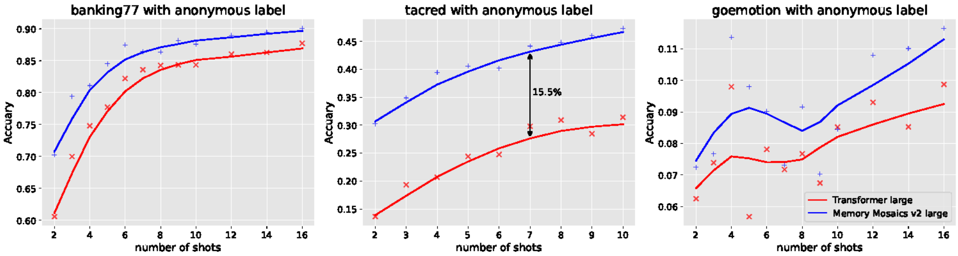

Figure 4: Anonymous label in-context learning comparison between memory mosaics v2 and transformers. Memory mosaics v2 significantly outperform transformers on all classification tasks.

In summary, the experiments demonstrate that memory mosaics v2 not only outperform transformer by a significant margin (more than 10%) on in-context learning, but also consistently improve performance as more demonstrations are provided. These results highlight the superior in-context learning ability of Memory Mosaics v2.

Augment transformer with long-short term attention Memory mosaics (v2) contains several unique components that are not applicable to transformers, such as the symmetric key and query, and the adaptive bandwidth. One seemingly applicable component for transformers is the separation of long-term and short-term memories introduced in Section 3.3. However, Figure 5 shows that augmenting a transformer with long-short-term attention does not help it overcome the limitations of in-context learning. These phenomena imply that memory mosaics (v2) is not simply a transformer variation but represents a different architecture.

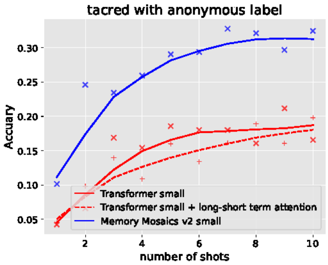

Figure 5: Augmenting transformer with long-short term attention doesn't help in-context learning.

Computation and Parameter Concerns On the last two evaluation dimensions (new knowledge storage and retrieval, and in-context learning), memory mosaics v2 outperform transformers by more than 10% with slightly more parameters. This 10% advantage holds even when comparing under the same number of parameters or the same computational budget. See Appendix Figure 12 for details.

6 Risk-return trade-off of frontier-model-sized memory mosaics v2

Having demonstrated the superior new tasks learning ability of memory mosaics v2 up to 9.9 billion parameters and 1 trillion training tokens, this section analyzes the "risk-return trade-off" to further scale memory mosaics v2 to the size of the frontier model, unveiling potential benefits and challenges.

Two Approaches To train a large frontier foundational model, one can either:

1) take a low-risk-low-return approach by investing more resources (GPUs and data) and reusing old recipes (e.g. architecture), or
2) take a middle-risk-high-return approach by trying new smart techniques.

Taking the first approach, one can take advantage of existing software, hardware, experiences, and datasets to quickly "reproduce" a huge foundational model. However, this approach is unlikely to result in a model that stands out from others, as it is based on shared recipes.

In contrast, taking the latter approach may require optimizing software and hardware, adapting techniques, a sharp sense of research direction, and possessing a keen sense of research direction along with strong problem solving abilities.[14] Despite the high requirements for personnel, this approach holds the potential for tremendous breakthroughs.

Ultimately, the decision between these two approaches depends on the available resources and personnel. To aid in this decision-making process, this section provides a simple and brutal comparison:

> *How much more data does the transformer recipe approach need to match the performance of memory mosaics v2?*

6.1 Comparison of two approaches

To answer this question, we compare the new tasks learning ability[15] of memory mosaics v2 and transformers trained on various amounts of data. Specifically, multiple transformer models are trained on 200B, 1T, and 8T training tokens, while a memory mosaics v2 is trained on 1T training tokens.

New-knowledge storage and retrieval Table 5 shows the comparison on the new-knowledge storage and retrieval ability. Training on the same number of tokens (1T), transformers lag behind memory mosaics v2 by 12.3% (41.1% vs 53.4%). ×8 times more training tokens (8T) improves the performance of transformers. However, the resulting transformer (trained on 8T tokens) still lags behind memory mosaics v2 (trained on 1T tokens) by 6.5% (46.9% vs 53.4%).

Although further increasing training data may improve the performance of transformers in this evaluation dimension, it comes at the cost of significantly larger training cost (time and resource). Moreover, a serious problem occurs: we are running out of data!

Table 5: Comparison of memory mosaics v2 and transformers, trained on 4k and fine-tuned on 32k context length, on RULER question-answer tasks. ("transformer large*" uses group-query attention to reduce memory cost, increases training context length to 8k to boost long-context performance.)

model	context length	train tokens	4k	8k	16k	task-length 32k	64k
transformer large	32k	200B	48.6	42.9	40.7	33.8	×
transformer large	32k	1T	51.2	48.8	44.7	41.1	×
transformer large*	32k	8T	59.2	54.5	50.9	46.9	×
memory mosaics v2 large	32k	1T	58.9	55.5	54.9	53.4	46.4

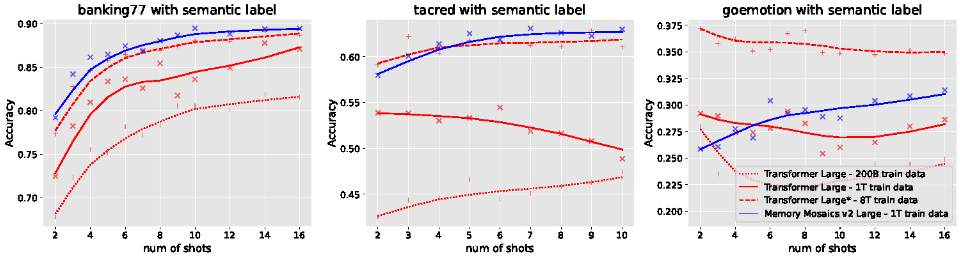

Figure 6: Semantic label in-context learning comparison between memory mosaics v2 and transformer. Memory mosaics v2 is trained on 1T tokens, while three transformers are trained on 200B, 1T, 8T tokens, respectively. Transformer with ×8 times more training data (8T, dash red line) starts to match the performance of Memory Mosaics v2 (1T, solid blue line).

---

[14] These requirements, in turn, demand a small group of high-quality researchers and managers.

[15] In i.i.d. regime, such as persistent-knowledge storing and retrieval, of course, more data + larger model = better performance. This argument in i.i.d. scenario has been well studied three decades ago Vapnik [1991].

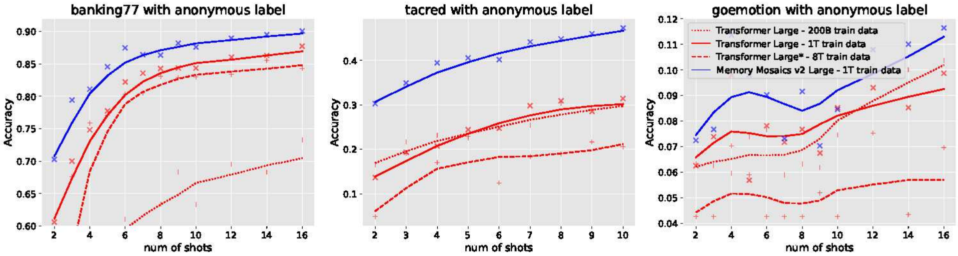

Figure 7: nonymous label in-context learning comparison between memory mosaics v2 and transformers. Sadly, transformers trained on 8T (dash red line) still lag behind memory mosaics v2 trained on 1T (solid blue line) by a large margin.

In-context learning Figures 6 and 7 show the comparison on in-context learning ability. For semantic label tasks (Figure 6), ×8 times more training data helps transformers (8T data) match the performance of memory mosaics v2 (1T data). However, for the more challenging anonymous label tasks, more training data cannot help transformers. Contour-intuitively, transformers trained on more training data (8T) exhibit a degraded performance on anonymous label tasks (Figure 7).

In summary, ×8 more training data helps transformers in certain new task learning benchmarks. However, the resulting transformers (8T data) still lag behind memory mosaics v2 trained on 1T data. More importantly, in anonymous label tasks that heavily rely on the new task learning ability, more training data cannot help transformers. These experiments answer the initial question: *"How much data does the transformer recipe approach need to match the performance of memory mosaics v2?"*.

7 Fine-tuning speed: who can fine-tune with one minibatch?

Despite the strong in-context learning capability of memory mosaics v2 shown in Section 5.3, it may still be attractive to fine-tune a model for a specific domain in order to either reduce inference costs or improve in-domain performance. It is generally expected that such models can be efficiently fine-tuned for a new domain using a comparatively small number of examples.

Figure 8 compares the fine-tuning speed (in terms of data size) of memory mosaics v2 and transformers. Both models, pre-trained on 4k context windows, were fine-tuned to 32k context length using the recipe described in Section 4 and evaluated on the same RULER tasks (32k task-length) described in Section 5.2.

Surprisingly, a single fine-tuning mini-batch (one optimization step) on memory mosaics v2 yields a 22% accuracy improvement. Two fine-tuning mini-batches on memory mosaics v2 are sufficient to reach the optimal performance. In contrast, a transformer fine-tuned with 800 mini-batches still lags behind memory mosaics v2 fine-tuned with a single mini-batch.

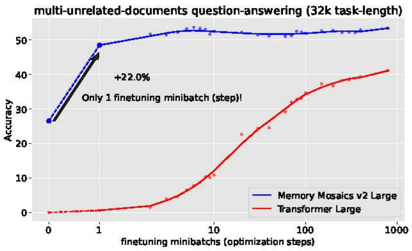

Figure 8: Fine-tuning speed comparison between memory mosaics v2 and transformer.

8 Discussion and future direction

This work scales memory mosaics (named memory mosaics v2) to llama-8B scale, demonstrating superior performance on new task learning, outperforming transformers by more than 10%. The three evaluation dimensions introduced in this work provide a transparent and controlled assessment of model capabilities, particularly focusing on the new task learning. The risk-return trade-off analysis reveals the weakness of the mainstream "more data more computation" belief, highlighting research opportunities on other smart techniques. One future direction is to reduce the computational cost for very long context lengths using fuzzy hashing Breitinger et al. [2014], Chen et al. [2024] and hierarchical memory Yuan et al. [2025], Lu et al. [2025] approaches.

Acknowledgments

Léon Bottou is a CIFAR fellow. We thank Gabriel Synnaeve, Jade Copet, Badr Youbi Idrissi, and Ammar Rizvi for their considerable support with hardware, software, data, and baselines.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2025.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36: 1560–1588, 2023.

Frank Breitinger, Barbara Guttman, Michael McCarrin, Vassil Roussev, and Douglas White. Approximate matching: Definition and terminology. techreport nist special publication 800-168. national institute of standards and technology, 2014.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*, 2020.

Adam Casson. Transformer flops. 2023. URL `https://adamcasson.com/posts/transformer-flops`.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. Magicpig: Lsh sampling for efficient llm generation. *arXiv preprint arXiv:2410.16179*, 2024.

Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

E. García-Portugués. *Notes for Nonparametric Statistics*. 2024. URL `https://bookdown.org/egarpor/NP-UC3M/`. Version 6.9.1. ISBN 978-84-09-29537-1.

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

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Vipul Gupta, David Pantoja, Candace Ross, Adina Williams, and Megan Ung. Changing Answer Order Can Decrease MMLU Accuracy, June 2024.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, et al. The elements of statistical learning, 2009.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Gregory Kamradt. Needle in a haystack - pressure testing llms. *Github*, 2023. URL `https://github.com/gkamradt/LLMTestNeedleInAHaystack/tree/main`.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models, October 2024.

E. Nadaraya. On estimating regression. *Theory of Probability and Its Applications*, 9:141–142, 1964.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Karsten Roth, Mark Ibrahim, Zeynep Akata, Pascal Vincent, and Diane Bouchacourt. Disentanglement of correlated factors via hausdorff factorized support. *arXiv preprint arXiv:2210.07347*, 2022.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Jianyu Zhang. Ai for the open-world: the learning principles. *arXiv preprint arXiv:2504.14751*, 2025.

Jianyu Zhang, Niklas Nolte, Ranajoy Sadhukhan, Beidi Chen, and Léon Bottou. Memory mosaics. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45, 2017.

Memory Mosaics at scale

Supplementary Material

A Gated time-variant key feature extractor & convolutional value extractor

Figure 9 illustrates how keys and values are constructed in memory mosaics v2.

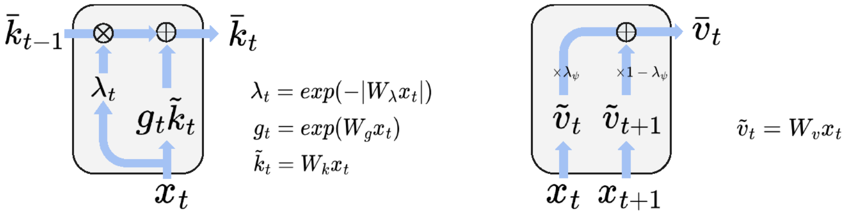

Figure 9: Left: Key k_t feature extractor: $k_t = \text{Norm}(\bar{k}_t)$. **Right:** Value v_t feature extractor: $v_t = \alpha_\psi \text{Norm}(\bar{v}_t)$.

B Training data sequence length distributions

Figure 10 shows the distributions of the length of the training data sequence truncated to 4096 or 32,768 max length.

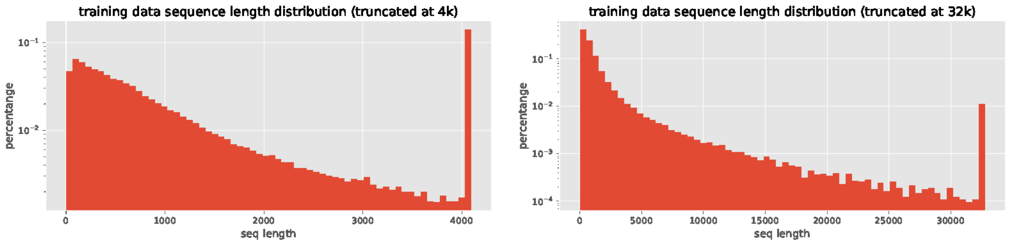

Figure 10: Training data sequence length distributions. For a given maximum sequence length during training (e.g. 4k), longer sequences are truncated to the maximum sequence length. This truncation results in the peaks at the end of distributions.

C Training details

hyperparameters For all Memory Mosaics v2 and baseline Transformer models[16], we use a consistent set of hyperparameters. That is, a batch size of 1024, a sequence length of 4096, an adamw optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ accompanied by a L_2 weight decay of 0.1 and a gradient norm clip of 1, a learning rate warm-up of 2000 iterations followed by a cosine learning rate scheduler that reduces the learning rate by a factor of 100 at the end. The initial learning rates (after warm-up) are set to 3e-4 for "small" models and 1e-3 for "large" models.

We also employ document-wise attention mask, where the attention scores are only computed within each sequence (document) in the training data, to reduce computation cost. Two special tokens, "<|begin_of_text|>" and "<|end_of_text|>" are appended at the begining and ending of a sequence, respectively.

During training, memory mosaics v2 samples the long-term memory delay step m from $[64, 256]$, sets the short-term memory window size $h = 256$. At inference, m is set to 64, as illustrated in Figure 11.

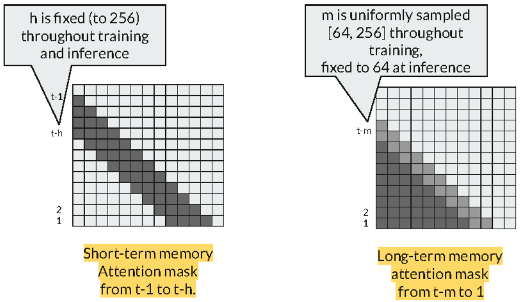

Figure 11: Randomly overlapped long-term & short-term memory.

It is worth noting that these hyperparameters were originally searched and optimized for the baseline transformer models. We transfer these hyperparameters to memory mosaics v2 without further hyperparameter searching. Thus, it is possible that this hyperparameter setup is suboptimal for memory mosaics v2.

Parameter Initialization and reparameterization Table 6 summarizes the parameter initialization methods and reparameterization tricks. W_1, W_2, W_3 refer to the parameters in persistent memory that are implemented as two-layer dense neural networks, $W_2\big(SiLU(W_1(x)) * W3(x)\big)$. $SiLU(x) = x \cdot sigmoid(x)$ is an activation function. $d \in \{2048, 4096\}$ indicates the hidden dimension of Memory Mosaics v2 small and large. $d' \in \{6144, 14336\}$ indicates the hidden dimension of the two-layer neural networks in persistent memory. l indicates the depth of the Mosaics blocks, starting from 0.

Table 6: Parameter initialization methods and reparameterization tricks used in Memory Mosaics v2.

Parameter	Location	reparameterization	Initialization
β_0	adaptive bandwidth	$\beta_0 = e^{\min(\theta,10)}$	$\theta = 1.5$
β_1	adaptive bandwidth	$\beta_1 = e^{\min(\theta,10)}$	$\theta = 1.5$
α	adaptive bandwidth	$\alpha = \min(\lvert\theta\rvert, 1)$	$\theta = 1/3$
α	feature extractor	$\alpha = e^{\min(\lvert\theta\rvert,15)}$	$\theta = 0$