FORGETTING TRANSFORMER: SOFTMAX ATTENTION WITH A FORGET GATE

Anonymous authors
Paper under double-blind review

ABSTRACT

An essential component of modern recurrent sequence models is the *forget gate*. While Transformers do not have an explicit recurrent form, we show that a forget gate can be naturally incorporated into Transformers by down-weighting the unnormalized attention scores in a data-dependent way. We name the resulting model the *Forgetting Transformer*. We show that the Forgetting Transformer significantly outperforms the standard Transformer on long-context language modeling and downstream tasks. Moreover, the Forgetting Transformer does not require any position embeddings and generalizes beyond the training context length. Several analyses, including the needle-in-the-haystack experiment, show that the Forgetting Transformer also retains the standard Transformer's superior long-context capabilities over recurrent sequence models such as Mamba-2, HGRN2, and DeltaNet.

1 INTRODUCTION

Despite the growing interest in reviving recurrent sequence models (Gu et al., 2021; Peng et al., 2021; Yang et al., 2023; Gu & Dao, 2023; Sun et al., 2023; De et al., 2024; Qin et al., 2024b; Dao & Gu, 2024; Peng et al., 2024; Beck et al., 2024; Zhang et al., 2024), these models still underperform the Transformer (Vaswani et al., 2017) in terms of *long-context capabilities* (Hsieh et al., 2024; Waleffe et al., 2024; Shen et al., 2024; Qin et al., 2024a), likely due to their relatively small fixed-sized hidden states (Jelassi et al., 2024). While the Transformer excels in handling long-context information, it lacks an explicit mechanism for forgetting past information in a *data-dependent* way[1]. Such a mechanism – often implemented as some form of the *forget gate* (Gers et al., 2000) – is ubiquitous in recurrent sequence models and has proven critical in their success in short-context tasks (Greff et al., 2016; Van Der Westhuizen & Lasenby, 2018; Peng et al., 2021; Yang et al., 2023; Gu & Dao, 2023). A natural question to ask is then: can we have a forget gate in Transformers?

To address this question, we leverage an important fact: many recurrent sequence models with a forget gate can be written in a parallel linear attention form (Katharopoulos et al., 2020) analogous to softmax attention (Yang et al., 2023; Dao & Gu, 2024). In this parallel form, the forget gate mechanism translates into down-weighing the unnormalized attention scores in a data-dependent way. Our key insight is that this *exact* mechanism is also applicable to softmax attention. We name the resulting model the *Forgetting Transformer*.

We evaluate the Forgetting Transformer on long-context language modeling and downstream tasks and find it significantly outperforms the standard Transformer. It also combines the strengths of both recurrent sequence models and the Transformer. Like recurrent sequence models, the Forgetting Transformer generalizes beyond the training context length, where the standard Transformer fails completely. At the same time, it retains the ability of the Transformer to perform accurate long-context retrieval and achieves perfect accuracy within the training context length in a simplified needle-in-the-haystack test (Kamradt, 2023). In contrast, all the tested recurrent sequence models fail. The Forgetting Transformer even achieves perfect retrieval up to *double* the training context length, demonstrating both accurate long-context retrieval and length generalization. Finally, we

[1]In principle, the Transformer can ignore previous information by generating keys with low dot-product values with all previous queries. However, this may not be as effective as an explicit forget gate. Also, certain methods such as AliBi (Press et al., 2021) achieve data-*independent* decay, as we will discuss later.

show that the Forgetting Transformer can be implemented in a hardware-aware way with a modified Flash Attention (Dao, 2023) algorithm.

2 Background: Linear attention with a forget gate

This section introduces the notation used in this work and gives a brief background on linear attention. We also introduce a gated variant of linear attention and discuss its parallel form, which naturally leads to the Forgetting Transformer. Throughout this work, we only consider causal sequence modeling. We also mainly consider the single-head case; extension to the multi-head case is straightforward.

2.1 Linear Attention

Standard causal softmax attention takes a sequence of input vectors $(\boldsymbol{x}_i)_{i=1}^L$ and produces a sequence of output vectors $(\boldsymbol{o}_i)_{i=1}^L$, where $\boldsymbol{x}_i, \boldsymbol{o}_i \in \mathbb{R}^d, i \in \{1, \dots, L\}$. Each $\boldsymbol{o}_i$ is computed as follows:

$$\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i = \boldsymbol{W}_q \boldsymbol{x}_i, \boldsymbol{W}_k \boldsymbol{x}_i, \boldsymbol{W}_q \boldsymbol{x}_i \in \mathbb{R}^d, \tag{1}$$

$$\boldsymbol{o}_i = \frac{\sum_{j=1}^i k_{\exp}(\boldsymbol{q}_i, \boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^i k_{\exp}(\boldsymbol{q}_i, \boldsymbol{k}_j)} = \frac{\sum_{j=1}^i \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^i \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j)}, \tag{2}$$

where $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times d}$ are projection matrices and $k_{\exp}(\boldsymbol{q}, \boldsymbol{k}) = \exp(\boldsymbol{q}^\top \boldsymbol{k})$ is the exponential dot product kernel. Note we omit the $\frac{1}{\sqrt{d}}$ scaling factor to reduce visual clutter. In practice we always scale the dot product $\boldsymbol{q}_i^\top \boldsymbol{k}_j$ by $\frac{1}{\sqrt{d}}$.

Linear attention (Katharopoulos et al., 2020) replaces the exponential dot product kernel $k_{\exp}(\boldsymbol{q}, \boldsymbol{k}) = \exp(\boldsymbol{q}^\top \boldsymbol{k})$ with a kernel $k_\phi(\boldsymbol{q}, \boldsymbol{k})$ with some feature representation $\phi : \mathbb{R}^d \to (\mathbb{R}^+)^{d'}$:

$$\boldsymbol{o}_i = \frac{\sum_{j=1}^i k_\phi(\boldsymbol{q}_i, \boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^i k_\phi(\boldsymbol{q}_i, \boldsymbol{k}_j)} = \frac{\sum_{j=1}^i (\phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j))\boldsymbol{v}_j}{\sum_{j=1}^i \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)}$$

Following Yang et al. (2023), we call the above the *parallel form* of linear attention as it can be computed with matrix multiplications. Alternatively, linear attention can be computed in a *recurrent form*:

$$\begin{aligned}
\boldsymbol{S}_t &= \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top \\
\boldsymbol{z}_t &= \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t) \\
\boldsymbol{o}_t &= \frac{\boldsymbol{S}_t \phi(\boldsymbol{q}_t)}{\boldsymbol{z}_t^\top \phi(\boldsymbol{q}_t)},
\end{aligned}$$

where $\boldsymbol{S}_t \in \mathbb{R}^{d \times d'}, \boldsymbol{z}_t \in \mathbb{R}^{d'}, t \in \{0, \dots, L\}$ are computed recurrently, with $\boldsymbol{S}_0 = \boldsymbol{0}$ and $\boldsymbol{z}_t = \boldsymbol{0}$.

2.2 Linear Attention with a forget gate

The recurrent form of linear attention makes it natural to introduce a forget gate. Specifically, we can compute a scalar forget gate $f_t = \sigma(\boldsymbol{w}_f^\top \boldsymbol{x}_t + b_f) \in \mathbb{R}$ at each timestep, where σ is the sigmoid function and $\boldsymbol{w}_f \in \mathbb{R}^d, b_f \in \mathbb{R}$ are learnable parameters. The recurrent computation is then:

$$\begin{aligned}
\boldsymbol{S}_t &= f_t \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top \\
\boldsymbol{z}_t &= f_t \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t) \\
\boldsymbol{o}_t &= \frac{\boldsymbol{S}_t \phi(\boldsymbol{q}_t)}{\boldsymbol{z}_t^\top \phi(\boldsymbol{q}_t)}.
\end{aligned}$$

Note that this gated variant of linear attention differs from most models in the literature. In particular, most gated variants of linear attention models such as GLA (Yang et al., 2023) and Mamba-2 (Dao & Gu, 2024) do not have the normalization term (i.e., there is no $\boldsymbol{z}_t$ and the output is just $\boldsymbol{o}_t = \boldsymbol{S}_t \phi(\boldsymbol{q}_t)$). We keep the normalization term to maintain similarity with softmax attention. The most similar model is gated-RFA (Peng et al., 2021), with the only difference being the lack of a $(1 - f_t)$ term in the recurrence.

Crucially, similar to the normalization-free version derived in GLA and Mamba-2, we can show that this gated variant of linear attention also has a parallel form:

$$\boldsymbol{o}_i = \frac{\sum_{j=1}^{i} F_{ij}\phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^{i} F_{ij}\phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)} = \frac{\sum_{j=1}^{i} F_{ij} k_\phi(\boldsymbol{q}_i, \boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^{i} F_{ij} k_\phi(\boldsymbol{q}_i, \boldsymbol{k}_j)}, \tag{3}$$

where $F_{ij} = \prod_{l=j+1}^{i} f_l$.[2]

Our key observation is that Equation 3 and the softmax attention in Equation 2 are very similar in form. In fact, if we just change the kernel k_ϕ in Equation 3 back to the exponential dot product kernel $k_{\exp}$, we obtain the *softmax attention with a forget gate*. We introduce this formally in the next section.

3 Forgetting Transformer

Our proposed model, the *Forgetting Transformer* (abbreviated as *ForT* in figures and tables), features a modified softmax attention mechanism with a forget gate. We name this attention mechanism the *Forgetting Attention*.

Forgetting Attention modifies the computation of the attention scores in softmax attention. Similar to the gated variant of linear attention introduced in the previous section, we compute a scalar forget gate $f_t = \sigma(\boldsymbol{w}_f^\top \boldsymbol{x}_t + b_f) \in \mathbb{R}$ for each timestep t. The output of the attention is then

$$\boldsymbol{o}_i = \frac{\sum_{j=1}^{i} F_{ij} \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^{i} F_{ij} \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j)} = \frac{\sum_{j=1}^{i} \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j + d_{ij})\boldsymbol{v}_j}{\sum_{j=1}^{i} \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j + d_{ij})} \tag{4}$$

where $F_{ij} = \prod_{l=j+1}^{i} f_l$ and $d_{ij} = \log F_{ij}$. This can be written in matrix form:

$$\boldsymbol{D} = \log \boldsymbol{F} \in \mathbb{R}^{L \times L}, \tag{5}$$

$$\boldsymbol{O} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top + \boldsymbol{D})\boldsymbol{V} \in \mathbb{R}^{L \times d}, \tag{6}$$

where $\boldsymbol{F} \in \mathbb{R}^{L \times L}$ is a lower triangular matrix whose non-zero entries are F_{ij}, i.e., $\boldsymbol{F}_{ij} = F_{ij}$ if $i \geq j$ and 0 otherwise. We adopt the convention that $\log 0 = -\infty$. $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{O} \in \mathbb{R}^{L \times d}$ are matrices containing $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i, \boldsymbol{o}_i, i \in \{1, \ldots, L\}$ as the rows. The $\mathrm{softmax}$ operation is applied row-wise.

The above describes the single-head case. For multi-head attention with h heads, we maintain h instances of forget gate parameters $(\boldsymbol{w}_f^{(i)})_{i=1}^{h}$ and $(b_f^{(i)})_{i=1}^{h}$ and compute the forget gates separately for each head.

Hardware-aware implementation Directly computing the attention output according to Equation 6 requires instantiating several $L \times L$ matrices in the slow high-bandwidth memory (HBM) of GPUs, which is extremely inefficient. Fortunately, the logit bias form on the rightmost side of Equation 4 allows the Forgetting Attention to be computed with a simple modification to the Flash Attention (Dao, 2023) algorithm.

Here we briefly describe the forward pass. The backward pass follows a similar idea. First, we compute the cumulative sum $c_i = \sum_{l=1}^{i} \log f_l$ for $i \in \{1, \ldots, L\}$ and store it in HBM. Note that this allows us to compute $d_{ij} = c_i - c_j$ easily later. Whenever we compute the attention logit via the dot product $\boldsymbol{q}_i^\top \boldsymbol{k}_j$ in the GPU's fast shared memory (SRAM) (as in Flash Attention), we also load c_i and c_j to SRAM, compute d_{ij}, and add the bias to the attention logit. The rest of the forward pass remains the same as Flash Attention.

This algorithm avoids instantiating the $L \times L$ d_{ij} entries on HBM. We provide a detailed algorithm description in Appendix C. Moreover, since the forget gates are scalars instead of vectors, the additional computation and parameter count introduced are negligible.

[2]We adopt the convention that $F_{ij} = 1$ if $i = j$.

Connection to ALiBi Besides its natural connection to gated linear attention, the Forgetting Attention can also be seen as a data-dependent and learnable version of ALiBi (Press et al., 2021). ALiBi applies a data-independent bias $b_{ij} = -(i-j)m_h$ to the attention logits, where m_h is a fixed slope specific to each head h. It is easy to show that ALiBi is equivalent to Forgetting Attention with a fixed, head-specific, and data-independent forget gate $f_t = \exp(-m_h)$. In Section 4.5, we verify the superiority of Forgetting Attention over ALiBi-based attention.

Position embeddings Though we find that using Rotary Position Embeddings (RoPE) (Su et al., 2024) improves the performance of the Forgetting Transformer *within the training context length*, it is not necessary as it is for the standard Transformer. More importantly, we find that RoPE damages generalization beyond the training context length. Therefore, *we do not use RoPE or any other position embeddings for the Forgetting Transformer by default*. This topic is studied in more detail in Section 4.5.

Architecture design Forgetting Attention can be used as a drop-in replacement for standard softmax attention in any Transformer architecture. Since architecture design is not the focus of this work, our Forgetting Transformer models use the same architecture as LLaMA (Touvron et al., 2023), except that we replace standard attention with Forgetting Attention and we do not use RoPE. However, similar to the findings in Dehghani et al. (2023), we find it helpful to apply RMSNorm (Zhang & Sennrich, 2019) to the queries and keys (i.e., QK-norm) in some tasks, so we also include results with QK-norm. Whether QK-norm is used in each result will be clearly stated.

4 Empirical study

The advantage of Transformers in long-context capabilities over recurrent sequence models have been demonstrated multiple times (Hsieh et al., 2024; Waleffe et al., 2024; Shen et al., 2024; Qin et al., 2024a). However, a forget gate introduces a *recency bias*. It is thus natural to ask whether the Forgetting Transformer still maintains this advantage. Therefore, our empirical study places a special focus on long-context capabilities.

4.1 Experimental setup

Dataset We focus on long-context language modeling and train all models on LongCrawl64 (Buckman, 2024). LongCrawl64 is a filtered long-sequence subset of RedPajama-v2 (Together Computer, 2023). It consists of pre-tokenized sequences truncated to exactly 64 kibitokens (KiT).[3] The sequences are tokenized with the TikToken tokenizer (OpenAI, 2022) for GPT-2 (Radford et al., 2019).

Baselines We are interested in two types of comparisons. First, to understand the benefits of forget gates, we compare our proposed model with the standard Transformer. Both the Transformer and the Forgetting Transformer use the LLaMA architecture, except that the Forgetting Transformer does not use RoPE. Similar to Xiong et al. (2023), we find it crucial to use a large RoPE angle θ for the standard Transformer. Following Xiong et al. (2023) we use $\theta = 500000$. As mentioned in Section 3, we test the Forgetting Transformer both with and without QK-norm. Note the comparison between the standard Transformer and the Forgetting Transformer (without QK-norm) is strictly controlled in that they only differ in whether they use the F_{ij} factors or RoPE.

Second, to demonstrate the advantage of the Forgetting Transformer over recurrent sequence models in long-context capabilities, we compare with Mamba-2 (Dao & Gu, 2024), HGRN2 (Qin et al., 2024a), and DeltaNet (Yang et al., 2024). These models are representative of various design choices in recurrent sequence models. Notably, all of them have reported better performance over the Transformer in terms of language modeling perplexity and mostly *short-context* downstream tasks. The implementation of all models is based on the Flash Linear Attention repository (Yang & Zhang, 2024).

[3]The binary prefix "kibi" or "Ki" means $2^{10} = 1024$. So 64 KiT means 65536 tokens. In the following we also use "mebi" or "Mi" for 2^{20} and "gibi" or "Gi" for 2^{30}.

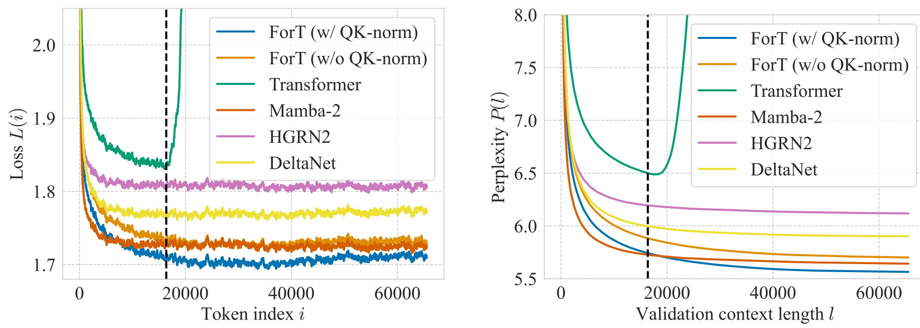

Figure 1: (**left**) Per-token loss $L(i)$ at different token position i. (**right**) Validation perplexity $P(l)$ over different validation context length l. The vertical dashed line indicates the training context length. The per-token loss is typically noisy, so we smooth the curve using a moving average sliding window of 101 tokens.

Training setup Due to limited compute resources, for our main experiments, we train models with 760M non-embedding parameters on a 15-GiT (roughly 16B tokens) subset of LongCrawl64 with a training context length of 16384 tokens. This roughly matches the compute-optimal model size/data ratio in Chinchilla scaling law (Hoffmann et al., 2022). For the validation set, we use a 2-GiT (roughly 2.1B tokens) subset of the LongCrawl64 held-out set consisting of sequences of 65536 tokens. We choose a much longer validation context length than the training context length to test the length generalization capabilities of the models.

All models are trained with AdamW (Loshchilov, 2017) with $(\beta_1, \beta_2) = (0.9, 0.95)$. We use a linear learning rate warmup from 0 to 1.25×10^{-3} for the first 256 MiT and then a cosine decay schedule to 1.25×10^{-4}. All models use a weight decay of 0.1 and gradient clipping of 1.0. We use `bfloat16` mixed-precision training for all models. More details of the experimental setup can be found in Appendix A.

4.2 Long-context language modeling

Metrics Before we present our results, it is important to understand one of our main metrics: *per-token* loss on the validation set at different token positions. To be precise, let V be the vocabulary size, $\boldsymbol{y}_i^{(j)} \in \{0,1\}^V$ be the one-hot vector encoding the language modeling target for the i-th token in the j-th validation sequence, and $\boldsymbol{p}_i^{(j)} \in \mathbb{R}^V$ be the corresponding output probabilities of the model, then the per-token loss $L(i)$ at token position i is simply

$$L(i) = \frac{1}{M} \sum_{j=1}^{M} -\log[(\boldsymbol{p}_i^{(j)})^\top \boldsymbol{y}_i^{(j)}], \tag{7}$$

where M is the number of validation sequences.

The per-token loss is particularly meaningful for understanding the long-context capabilities of a model. Informally, a monotonically decreasing $L(i)$ with a steep slope indicates the model is using the full context well. On the other hand, if $L(i)$ plateaus after some token position k, it indicates the model is incapable of using tokens that are k tokens away from the current token position for its prediction. This correspondence between the slope of $L(i)$ and the model's context utilization is explained in more detail in Appendix B.

Besides per-token loss, we also report perplexity over different validation context lengths $P(l)$. Specifically, perplexity over a context length l is defined as $P(l) = \exp(\frac{1}{l}\sum_{i=1}^{l} L(i))$. We warn the readers that the slope of $P(l)$ is less meaningful. Since $P(l)$ is just the exponential of the cumulative average of $L(i)$, even if $L(i)$ plateaus after some token position k, $P(l)$ will still monotonically decrease after k, giving the wrong impression that the model can make use of the part of the context that is k tokens away.

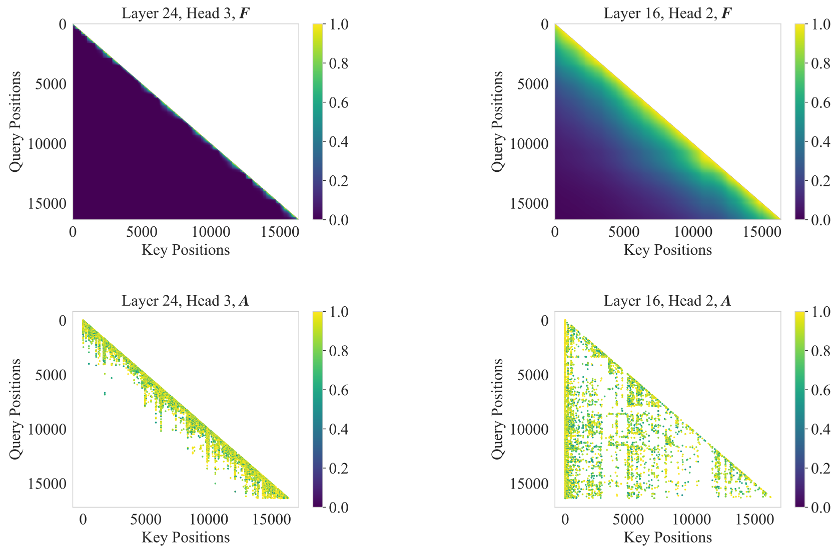

Figure 2: Visualization of the forget gate weight matrix $\boldsymbol{F}$ (top row) and the attention score matrix $\boldsymbol{A}$ (bottom row) from two heads in different layers. Since $\boldsymbol{A}$ is very sparse, we only show entries with scores larger than 0.5. These results use a Forgetting Transformer without QK-norm.

Results In Figure 1, we show the per-token loss $L(i)$ at different token indices i and perplexity $P(l)$ over different validation context lengths l. As shown in Figure 1, with or without QK-norm, the Forgetting Transformer (shown as *ForT*) significantly outperforms the standard Transformer. Similarly to the standard Transformer, it also maintains a monotonic decreasing per-token loss *within the training context length*, indicating that it utilizes the entire training context for its prediction. In contrast, the per-token loss curves of all recurrent sequence models start flattening at around 5k tokens and completely plateau after 10k tokens. This indicates that these recurrent sequence models struggle to use the full context effectively for their prediction.

The Forgetting Transformer also generalizes beyond the training context length, where the standard Transformer fails completely. In terms of the *absolute* values of the loss, the Forgetting Transformer also clearly outperforms HGRN2 and DeltaNet, and outperforms Mamba-2 at later tokens when QK-norm is used.

Visualization of forget gate values and attention map In Figure 2, we visualize the forget gate weight matrix $\boldsymbol{F}$ and the attention scores $\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top + \log \boldsymbol{F})$ from two heads in different layers. The head on the left-hand side exhibits strong decay, and most entries of $\boldsymbol{F}$ are close to zero; accordingly, the attention focuses on local entries. The head on the right-hand side has much weaker decay, and the attention is distributed across the entire context. This shows that the Forgetting Transformer can learn to retain information across long contexts when necessary.

4.3 Needle in the Haystack

The needle-in-the-haystack analysis (Kamradt, 2023) (referred to as the "needle test" in the following) is a popular test for the long-context retrieval abilities of language models. Following Qin et al. (2024a), we use an "easy mode" of the needle test, where the "needle" placed within the context includes both the question and the answer. This easy mode is particularly suitable for base models that have not been instruction-tuned. Full details, including the prompts used, are in Appendix A.2.

In Figure 3, we show the results of the needle test for the Transformer, the Forgetting Transformer (with and without QK-norm), and Mamba2. DeltaNet and HGRN2's results are even worse than Mamba-2, so we leave them to Appendix D.2. We use sequences of up to 32000 tokens for the test, which is almost double the training context length 16384. As shown in Figure 3, both the Forgetting Transformer and the Transformer achieve near-perfect needle retrieval within the training context

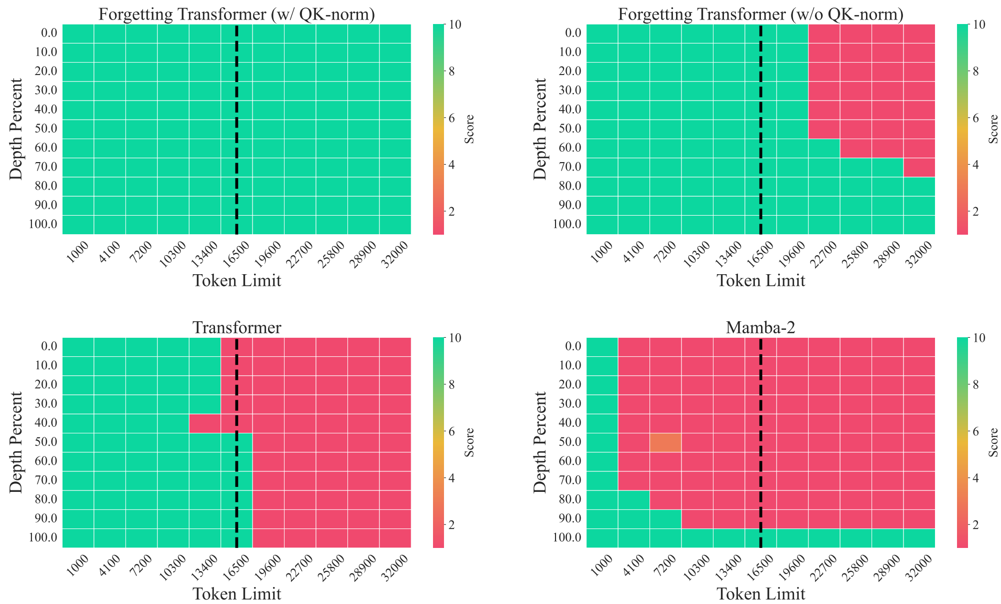

Figure 3: Needle-in-the-haystack analysis for different models. The results are scored on a scale of 1 (red) to 10 (green) by GPT-4o. The vertical dashed line indicates the training context length.

length. Interestingly, with QK-norm, the Forgetting Transformer even achieves perfect retrieval up to double the training context length[4], while the standard Transformer fails. In contrast, Mamba2 (and also HGRN2 and DeltaNet in Appendix D.2) fails even within the training context length, except when the needle is placed right at the end of the text. These results are consistent with the previous analysis of the slope of per-token loss curves in Section 4.2.

4.4 Downstream tasks

We evaluate the models on two sets of downstream tasks: a set of short-context tasks from LM-evaluation-harness (Gao et al., 2024) and a set of long-context tasks from LongBench (Bai et al., 2023).

Short-context tasks We use Wikitext (Merity et al., 2016), LAMBADA (Paperno et al., 2016), PiQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Zellers et al., 2019), ARC-easy, ARC-challenge (Clark et al., 2018), Copa (Roemmele et al., 2011), SciQA (Auer et al., 2023), OpenbookQA (Mihaylov et al., 2018), and BoolQA (Clark et al., 2019). Following Yang et al. (2023), we report perplexity for Wikitext and LAMBADA, length-normalized accuracy for HellaSwag, ARC-challenge, and OpenbookQA, and accuracy for all other tasks (we also report accuracy for LAMBADA). All results are zero-shot.

As shown in Table 1, the Forgetting Transformer outperforms the standard Transformer on almost all the tasks, with or without QK-norm. This demonstrates the effectiveness of a forget gate in the attention layer. The inclusion of a forget gate also allows the Forgetting Transformer to outperform DeltaNet and HGRN2, and performs on par with Mamba-2 on these short-context tasks.

Long-context tasks We use 14 tasks from LongBench: HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), MultiFieldQA-en, NarrativeQA (Kočiskỳ et al., 2018), Qasper (Dasigi et al., 2021), GovReport (Huang et al., 2021), QMSum (Zhong et al., 2021), MultiNews (Fabbri et al., 2019), TriviaQA (Joshi et al., 2017), SAMSum (Gliwa et al., 2019), TREC (Li & Roth, 2002), LCC (Guo et al., 2023), and RepoBench-P (Liu et al., 2023). These are all generation-based tasks with average lengths ranging from roughly 1k words to up to 18k words. We

[4] The reason for the effectiveness of QK-norm in this case in unclear. We leave it for a future investigation.

Table 1: Evaluation results on LM-eval-harness. All models have roughly 760M non-embedding parameters and are trained on roughly 16B tokens on LongCrawl64. "acc-n" means length-normalized accuracy. Bold and underlined numbers indicate the best and the second best results, respectively.

Model	Wiki. ppl↓	LMB. ppl↓	LMB. acc↑	PIQA acc↑	Hella. acc-n↑	Wino. acc↑	ARC-e acc↑	ARC-c acc-n↑	COPA acc↑	OBQA acc-n↑	SciQA acc↑	BoolQ acc↑	Avg ↑
ForT (w/o QK-norm)	32.89	**29.25**	**35.67**	60.28	31.89	**51.85**	44.87	24.91	64.00	**29.80**	75.90	61.50	**48.07**
ForT (w/ QK-norm)	**31.91**	29.65	35.47	**61.21**	32.16	50.75	45.54	24.06	62.00	25.80	75.60	58.04	47.06
Transformer	37.47	50.15	29.83	60.34	29.86	50.28	44.65	23.63	61.00	28.60	71.70	**61.80**	46.17
Mamba-2	33.11	42.74	26.80	60.77	**32.74**	51.46	45.71	23.29	**69.00**	28.40	**76.30**	60.80	47.53
HGRN2	39.27	31.87	33.46	60.12	31.56	49.96	**47.60**	23.55	63.00	27.20	73.70	42.97	45.31
DeltaNet	35.12	47.49	28.24	60.07	30.83	51.07	46.30	**25.26**	65.00	28.00	71.40	50.80	45.69

Table 2: Evalution results on LongBench. All models have roughly 760M non-embedding parameters and are trained on roughly 16B tokens on LongCrawl64. Bold and underlined numbers indicate the best and the second best results, respectively.

Model	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Code	
	NarrativeQA	Qasper	MFQA	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SamSum	LCC	RepoBench-P
ForT (w/o QK-norm)	**9.42**	**12.38**	18.85	**7.6**	11.57	4.34	**23.38**	9.47	8.56	**47.0**	17.04	6.39	11.13	14.78
ForT (w/ QK-norm)	6.69	11.64	**19.38**	5.56	9.32	**5.37**	21.39	9.04	8.04	39.0	**19.08**	**11.5**	10.41	14.2
Transformer	7.41	10.94	17.64	6.2	15.84	3.34	10.79	9.38	**12.53**	18.5	9.47	2.4	11.23	17.36
Mamba-2	6.63	8.93	16.93	6.39	**17.01**	3.43	6.89	**13.07**	7.64	11.5	11.64	1.44	15.72	10.38
HGRN2	6.09	7.98	13.26	4.9	12.23	3.06	6.64	9.76	7.54	17.5	12.46	1.06	11.19	16.28
DeltaNet	6.6	7.57	15.25	5.13	12.88	3.21	6.94	10.49	7.9	13.5	13.6	6.04	**17.52**	**18.43**

use the default metrics of LongBench for different tasks, which are either F1, Rough-L, accuracy, or edit similarity.

The results are shown in Table 2. With or without QK-norm, the Forgetting Transformer obtains the best or the second-best results on the majority of the tasks, verifying its superior long-context capabilities.

4.5 Ablations

We present two sets of ablation studies. First, we investigate the effects of RoPE, particularly its influence on length generalization. Second, we study the importance of using a forget gate that is data-dependent. For these experiments, we use smaller models with 125M parameters trained on roughly 2.6B tokens. To ensure that the experiments are strictly controlled, we do not use QK-norm in any of the experiments.

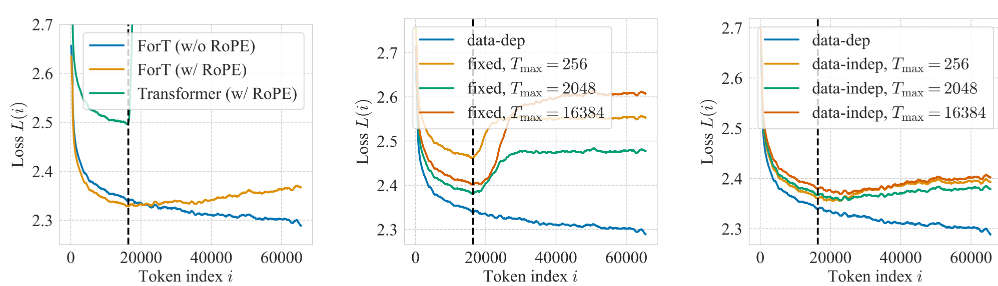

Figure 4: (**left**) Effect of adding RoPE. (**middle**) Data-dependent forget gate vs. fixed forget gate (i.e., ALiBI) (**right**) Data-dependent forget gate vs. data-independent forget gate. All per-token loss curves are smoothed by a moving average sliding window of 1001 tokens. The vertical dashed line indicates the training context length.

Effects of RoPE In Figure 4 (left) we show the per-token loss curve of three models: Transformer with RoPE, Forgetting Transformer without RoPE, and Forgetting Transformer with RoPE. We omit the Transformer without RoPE since it performs poorly (loss larger than 3.0). As shown in Figure 4, even though RoPE improves the performance of the Forgetting Transformer within the training context length, it damages length generalization beyond the training context length.

Data independent and fixed forget gates To show the importance of using a forget gate that is data-dependent, we test a data-*independent* forget gate $f_t^{(h)} = \sigma(b^{(h)})$, where the superscript (h) means for the h-th head. We also test a forget gate that has fixed values (i.e., $f_t^{(h)} = \sigma(b^{(h)})$, but we do not update $b^{(h)}$ during training). As discussed in Section 3, using a fixed forget gate is equivalent to ALiBi.

For these data-independent forget gate designs, we find it important to initialize $b^{(h)}$ properly. To understand the initialization, we first define a function $T(b) = \frac{1}{-\log \sigma(b)}$. This function is defined such that $\sigma(b)^{T(b)} = 1/e$ is always true. We then initialize $b^{(h)} = b_{\text{init}}^{(h)}$ such that $T(b_{\text{(init)}}^{(h)}) = \exp(\log T_{\min} + (\log T_{\max} - \log T_{\min})\frac{h-1}{H-1})$, where $T_{\min}$ and $T_{\max}$ are hyperparameters and H is the number of heads. It can be shown that ALiBi with a maximum slope $\frac{1}{2}$ and a minimum slope $\frac{1}{256}$ (the default values in Press et al. (2021)) is equivalent using a fixed forget gate with $(T_{\min}, T_{\max}) = (2, 256)$. We refer to this initialization as *long-init*. In the following experiments, we always set $T_{\min} = 2$. We also tested long-init for the data-dependent forget gate in Appendix D.1 but did not find it useful.

In Figure 4, we show the per-token loss of the Forgetting Transformer with a fixed forget gate (middle, shown as "fixed") and a data-independent forget gate (right, shown as "data-indep"). We also show the results with a data-dependent forget gate (shown as "data-dep") for comparison. As shown in Figure 4, a data-dependent forget gate works the best both within and beyond the training context length.

5 Related work

Recurrent sequence models While the Transformer has become the de facto standard architecture for sequence modeling, there has been a growing interest in reviving recurrent sequence models (Katharopoulos et al., 2020; Peng et al., 2021; Gu et al., 2021; Orvieto et al., 2023; Yang et al., 2023; Gu & Dao, 2023; Katsch, 2023; De et al., 2024; Sun et al., 2024; Peng et al., 2024; Qin et al., 2024a; Dao & Gu, 2024; Beck et al., 2024; Zhang et al., 2024; Buckman et al., 2024). Unlike traditional non-linear RNNs such as LSTMs (Beck et al., 2024) and GRUs (Chung et al., 2014), these models feature *linear recurrence* in the form $h_t = g(x_t)h_{t-1} + f(x_t)$, where x_t is the input, h_t is the (potentially matrix-valued) hidden state, and g, f are arbitrary functions. Besides its potential advantage for learning long-term dependencies (Orvieto et al., 2023), linear recurrence is also amenable to parallel computation (Martin & Cundy, 2017; Gu et al., 2021; Smith et al., 2022; Yang et al., 2023; Dao & Gu, 2024). Many recent recurrent sequence models feature some form of the *forget gate*, which has been shown to be essential in these architectures (Qin et al., 2024b; Gu & Dao, 2023; Yang et al., 2023). Notably, GLA (Yang et al., 2023) and Mamba-2 (Dao & Gu, 2024) show that gated variants of linear attention could be written in a form similar to softmax attention, which directly inspired our work.

Data-indepedent decay via position embeddings Several position embedding methods for Transformers achieve data-independent decay. ALiBi (Press et al., 2021), T5's RPE (Raffel et al., 2020), Kerple (Chi et al., 2022a), and Sandwich (Chi et al., 2022b) add bias to the attention logits depending on the distances between the keys and queries. When the bias is negative, this is equivalent to down-weighting previous timesteps. Though less explicit, RoPE (Su et al., 2024) also has a similar decay effect that becomes stronger with increasing relative query/key distances. However, all these methods can only achieve data-*independent* decay based on the relative distances of the queries and keys.

6 Conclusion

We propose the Forgetting Transformer, a Transformer variant with a forget gate. Our experiments show that the Forgetting Transformer outperforms the standard Transformers on both long-context and short-context tasks. The Forgetting Transformer also shows length generalization abilities beyond the training context length. We also propose a hardware-aware algorithm for Forgetting Transformer based on Flash Attention.

Our work has several limitations that present opportunities for future work. First, due to our limited computing resources, we can only perform experiments on models up to 760M parameters. Thus, an important future work is to extend the Forgetting Transformer to larger scales. Second, we do not investigate architectural design variations of the models (e.g., output gating and normalization as in Mamba-2), so there is likely still a large room for improvement in terms of performance. Finally, we only consider causal sequence modeling. It would be interesting to extend the Forgetting Transformer to the non-causal case.

References

Sören Auer, Dante AC Barone, Cassiano Bartz, Eduardo G Cortes, Mohamad Yaser Jaradeh, Oliver Karras, Manolis Koubarakis, Dmitry Mouromtsev, Dmitrii Pliukhin, Daniil Radyush, et al. The sciqa scientific question answering benchmark for scholarly knowledge. *Scientific Reports*, 13 (1):7240, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

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. *arXiv preprint arXiv:2405.04517*, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Jacob Buckman. Longcrawl64: A Long-Context Natural-Language Dataset, 2024. URL `https://manifestai.com/articles/longcrawl64`.

Jacob Buckman, Carles Gelada, and Sean Zhang. Symmetric Power Transformers, 2024.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander Rudnicky. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399, 2022a.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. Dissecting transformer length extrapolation via the lens of receptive field analysis. *arXiv preprint arXiv:2212.10356*, 2022b.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*, 2021.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512. PMLR, 2023.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*, 2019.

FlagOpen, 2023. URL `https://github.com/FlagOpen/FlagAttention`.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL `https://zenodo.org/records/12608602`.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*, 2019.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016.

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

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pp. 12098–12107. PMLR, 2023.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

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. *arXiv preprint arXiv:2203.15556*, 2022.

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

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*, 2021.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*, 2024.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Gregory Kamradt, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/blob/main/README.md.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Tobias Katsch. Gateloop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*, 2023.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems. *arXiv preprint arXiv:2306.03091*, 2023.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. *arXiv preprint arXiv:1709.04057*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

OpenAI, 2021. URL https://github.com/triton-lang/triton.

OpenAI, 2022. URL https://github.com/openai/tiktoken.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

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.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024a.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. *Advances in Neural Information Processing Systems*, 36, 2024b.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI spring symposium series*, 2011.

Xuyang Shen, Dong Li, Ruitao Leng, Zhen Qin, Weigao Sun, and Yiran Zhong. Scaling laws for linear complexity language models. *arXiv preprint arXiv:2406.16690*, 2024.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Together Computer. Redpajama: an open dataset for training large language models, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Jos Van Der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. *arXiv preprint arXiv:1804.04849*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL `https://github.com/sustcsonglin/flash-linear-attention`.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Jiajie Zhang Yuze He Ji Qi Lei Hou Jie Tang Yuxiao Dong Juanzi Li Yushi Bai, Xin Lv. Longalign: A recipe for long context alignment of large language models. *arXiv preprint arXiv:2401.18058*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *arXiv preprint arXiv:2409.07146*, 2024.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*, 2021.

A Experimental details

A.1 Model and training hyperparameters

All models in the main experiment have roughly 760M non-embedding parameters. We do not share the embedding parameters with the last linear layer. All models have a hidden dimension of 1536 and a head dimension of 128. As mentioned in the main text, we use $\theta = 500000$ for RoPE. For other hyperparameters, we use the default values in Flash Linear Attention (Yang & Zhang, 2024).

For the ablation experiments in Section 4.5, all models have roughly 125M non-embedding parameters. The hidden dimension is 768 and the head dimension is 64. For other model hyperparameters, we use the default values in Flash Linear Attention (Yang & Zhang, 2024). We use a linear learning rate warmup from 0 to 3×10^{-3} for the first 256 MiT and then a cosine decay schedule to 3×10^{-4}. Other training related hyperparameters are the same as the 760M-parameter setting.

A.2 Needle in the haystack details

We use the needle test in the LongAlign (Yushi Bai, 2024), which is adapted from the original needle test reposiroty (Kamradt, 2023) for HuggingFace[5] models. The prompt has the following structure:

```
[irrelevant context...]
What is the best thing to do in San Francisco? Answer: The best thing to
do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny
day.
[irrelevant context...]

There is an important piece of information hidden inside the above
document. Now that you've read the document, I will quiz you about it.
Answer the following question: What is the best thing to do in San
Francisco? Answer:
```

The results are scored by GPT-4o on a scale from 1 to 10.

B Explanation on the relationship between per-token-loss slope and context utilization

To understand the relationship between the slope of the per-token loss and context utilization of the model, we first point out that LongCrawl64 applies the preprocessing of randomly "rolling" the sequences[6] to remove any position bias. This means that *when given contexts of the same length*, the difficulty of predicting tokens at different positions is roughly the same on average. For example, predicting the 100-th tokens in the sequences *only given the previous* 90 *tokens* is roughly as difficult as predicting the 90-th tokens when given the full previous 90-token context. Therefore, if $L(100) < L(90)$, it indicates that the first 10 tokens in the context contribute to the model's predictions for the 100-th token; and larger the difference $L(90) - L(100)$ is, the more these distant tokens contribute. On the other hand, if $L(100)$ is roughly the same $L(90)$ (i.e., the graph of $L(i)$ plateaus after $i = 100$), it means the first 10 tokens do not contribute to the model's prediction for the 100-th token, either because they are inherently not useful for this prediction or the model are unable to utilize them.

In summary, the slope of $L(i)$ at token position i reflects how much tokens from roughly i steps earlier contribute to the model's prediction at the current token position.

C Hardware-aware implementation of Forgetting Attention

In Algorithm 1, we provide the algorithm for computing the forward pass of Forgetting Attention in a hardware-aware way. The algorithm is reproduced from Flash Attention 2 (Dao, 2023), with

[5] `https://huggingface.co/`

[6] Concretely, this can be implemented with `np.roll` with random shift value

the changes needed to implement Forgetting Attention added and highlighted. In this algorithm, we assume that we pre-computed the cumulative sum $\boldsymbol{c} = \mathrm{cumsum}(\boldsymbol{f})$, where $\boldsymbol{f} \in \mathbb{R}^N$ is a vector that stacks the N forget gates values across the sequence dimension N. In practice, we implement Forgetting Attention based on the Triton (OpenAI, 2021) Flash Attention implementation in Flag Attention (FlagOpen, 2023).

We omit the backward pass since the changes involved are basically the same as the forward passes, except that we also need to compute the gradients for $\boldsymbol{c}$ and then $\boldsymbol{f}$.

Algorithm 1 Forgetting Attention forward pass

Require: Matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N \times d}$, vector $\boldsymbol{c} \in \mathbb{R}^N$ in HBM, block sizes B_c, B_r.

Divide $\boldsymbol{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\boldsymbol{K}, \boldsymbol{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_{T_c}$ and $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{T_c}$, of size $B_c \times d$ each.
Divide the output $\boldsymbol{O} \in \mathbb{R}^{N \times d}$ into T_r blocks $\boldsymbol{O}_i, \ldots, \boldsymbol{O}_{T_r}$ of size $B_r \times d$ each, and divide the logsumexp L into T_r blocks $L_i, \ldots, L_{T_r}$ of size B_r each.
Let $\boldsymbol{c}^q = \boldsymbol{c}$. Devide $\boldsymbol{c}^q$ into T_r blocks $\boldsymbol{c}^q_1, \ldots, \boldsymbol{c}^q_{T_r}$
Let $\boldsymbol{c}^k = \boldsymbol{c}$. Devide $\boldsymbol{c}^k$ into T_c blocks $\boldsymbol{c}^k_1, \ldots, \boldsymbol{c}^k_{T_r}$
for $1 \le i \le T_r$ **do**
    Load $\boldsymbol{Q}_i$, $\boldsymbol{c}^q_i$ from HBM to on-chip SRAM.
    On chip, initialize $\boldsymbol{O}_i^{(0)} = (0)_{B_r \times d} \in \mathbb{R}^{B_r \times d}, \ell_i^{(0)} = (0)_{B_r} \in \mathbb{R}^{B_r}, m_i^{(0)} = (-\infty)_{B_r} \in \mathbb{R}^{B_r}$.
    for $1 \le j \le T_c$ **do**
        Load $\boldsymbol{K}_j, \boldsymbol{V}_j$, $\boldsymbol{c}^k_j$ from HBM to on-chip SRAM.
        On chip, compute $\boldsymbol{S}_i^{(j)} = \boldsymbol{Q}_i \boldsymbol{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
        On chip, compute $\boldsymbol{D}_i^{(j)} = \boldsymbol{c}^q_i \mathbf{1}^\top - \mathbf{1}(\boldsymbol{c}^k_j)^\top \in \mathbb{R}^{B_r \times B_c}$.
        On chip, compute $\boldsymbol{S}_i^{(j)} = \boldsymbol{S}_i^{(j)} + \boldsymbol{D}_i^{(j)} \in \mathbb{R}^{B_r \times B_c}$.
        On chip, compute $m_i^{(j)} = \max(m_i^{(j-1)}, \mathrm{rowmax}(\boldsymbol{S}_i^{(j)})) \in \mathbb{R}^{B_r}$, $\tilde{\mp}_i^{(j)} = \exp(\boldsymbol{S}_i^{(j)} - m_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\ell_i^{(j)} = e^{m_i^{j-1} - m_i^{(j)}} \ell_i^{(j-1)} + \mathrm{rowsum}(\tilde{\mp}_i^{(j)}) \in \mathbb{R}^{B_r}$.
        On chip, compute $\boldsymbol{O}_i^{(j)} = \mathrm{init}(e^{m_i^{(j-1)} - m_i^{(j)}})^{-1} \boldsymbol{O}_i^{(j-1)} + \tilde{\mp}_i^{(j)} \boldsymbol{V}_j$.
    end for
    On chip, compute $\boldsymbol{O}_i = \mathrm{init}(\ell_i^{(T_c)})^{-1} \boldsymbol{O}_i^{(T_c)}$.
    On chip, compute $L_i = m_i^{(T_c)} + \log(\ell_i^{(T_c)})$.
    Write $\boldsymbol{O}_i$ to HBM as the i-th block of $\boldsymbol{O}$.
    Write L_i to HBM as the i-th block of L.
end for
Return the output $\boldsymbol{O}$ and the logsumexp L.

D Additional results

D.1 Long-init for data-dependent forget gate

In Figure 5, we show the effect of using long-init for the data-dependent forget gate. As shown in the figure, long-init even damages performance.

D.2 Additional needle-in-the-haystack result

In Figure 6, we show the results of the needle test for HGRN2 and DeltaNet. Note they perform even worse than Mamba-2 shown in the main text.

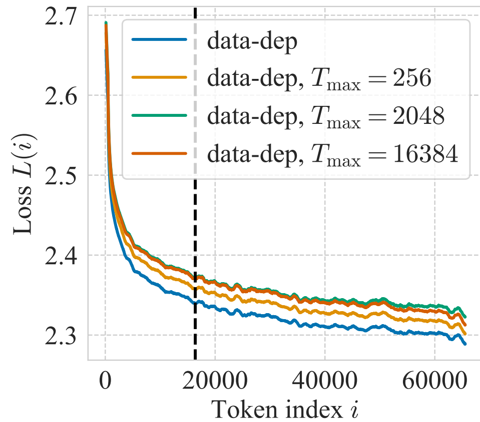

Figure 5: Using long-init for data-dependent forget gate. The per-token loss curve is smoothed with a moving average sliding window of 1001 tokens. The vertical dashed line indicates the training context length.

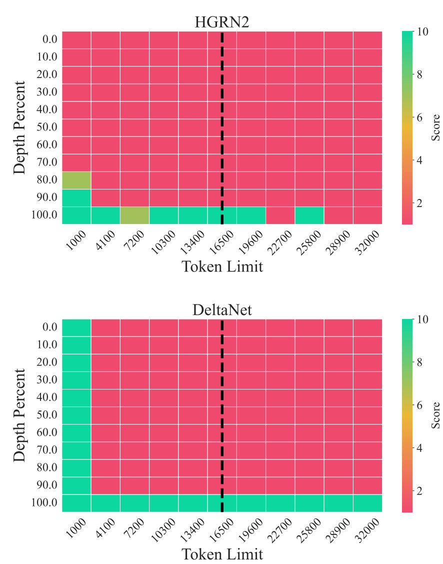

Figure 6: Needle-in-the-haystack analysis for HGRN2 and DeltaNet. The results are scored on a scale of 1 (red) to 10 (green). The vertical dashed line indicates the training context length.