TransformerFAM: Feedback attention is working memory

Dongseong Hwang [1] **Weiran Wang** [1] **Zhuoyuan Huo** [1] **Khe Chai Sim** [1] **Pedro Moreno Mengibar** [1]

Abstract

While Transformers have revolutionized deep learning, their quadratic attention complexity hinders their ability to process infinitely long inputs. We propose Feedback Attention Memory (FAM), a novel Transformer architecture that leverages a feedback loop to enable the network to attend to its own latent representations. This design fosters the emergence of working memory within the Transformer, allowing it to process indefinitely long sequences. TransformerFAM requires no additional weights, enabling seamless integration with pre-trained models. Our experiments show that TransformerFAM significantly improves Transformer performance on long-context tasks across various model sizes (1B, 8B, and 24B). These results showcase the potential to empower Large Language Models (LLMs) to process sequences of unlimited length.

1. Introduction

The introduction of the Transformer architecture (Vaswani et al., 2017) has revolutionized deep learning by permeating diverse domains and enhancing performance due to its efficacy and scalability. This scalability fuels a trend analogous to Moore's law, which links increased model size to performance gains (Kaplan et al., 2020).

The effectiveness of attention in text sequence processing was solidified through the Transformer paper. Models like BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020) further showcased the scalability of Transformer and its tendency for improved performance with increased model size. Following the replacement of LSTM (Hochreiter & Schmidhuber, 1997) by Transformer in most Natural Language Processing (NLP) domains, the Vision Transformer (ViT) (Dosovitskiy et al., 2020) replaced Convolutional Neural Network (CNN) (LeCun et al., 1995) with Transformers in the vision domain, and Conformer (Convolution-augmented Transformer) (Gulati et al., 2020) replaced

LSTM in the speech domain. The Transformer has become the de facto architecture in various domains. Currently, attention serves as the leading architecture for extracting meaningful representations from homogeneous data.

The logical progression points toward extending attention capabilities to heterogeneous data. This has enabled advances in multimodal fusion (text and vision), as seen in models like DALL·E 2 (Ramesh et al., 2022), Flamingo (Alayrac et al., 2022) and CoCa (Yu et al., 2022). AudioLM (Borsos et al., 2023) has shown that attention also excels at fusing audio and text. Consequently, Gemini (Team et al., 2023) integrates text, images, audio, and video into a single generative model. This was possible because attention to heterogeneous data works exceptionally well.

Despite the impressive success of attention, it suffers from major drawbacks. Firstly, attention has quadratic complexity with respect to context length, which limits the capability of modeling long contexts. Secondly, it forgets information from context before attention window, unlike LSTM, which theoretically can propagate information indefinitely. We want the better architecture to be able to process arbitrarily long sequences efficiently, while preserving very long-term dependencies.

Sliding window attention is introduced (Dai et al., 2019; Beltagy et al., 2020) to handle infinitely long sequences as input. However, it disregards information beyond the effective receptive field (approximately model depth × window size in Figure 1d) during sequence generation. Various approaches have attempted to address the long-context problem by sparse attention (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020; Roy et al., 2021; Oren et al., 2024) and linear approximated attention (Wang et al., 2020; Katharopoulos et al., 2020; Choromanski et al., 2020; Xiong et al., 2021), showing effectiveness below the 1B scale. Yet, scaling laws (Tay et al., 2022) suggest these approximations do not perform as well at GPT-3 level. Current SoTA large language model (LLM) leaderboards (LMSYS, 2023) do not feature architectures primarily relying on approximated attention.

On the other hand, Neuroscience links attention to multisensory integration (Tang et al., 2016). Endogenous (goal-driven) and exogenous (stimulus-driven) attention are distributed throughout sensory processing regions in brain, and

---

[1]Google LLC, Mountain View, CA, USA. Correspondence to: Dongseong Hwang <dongseong@google.com>.
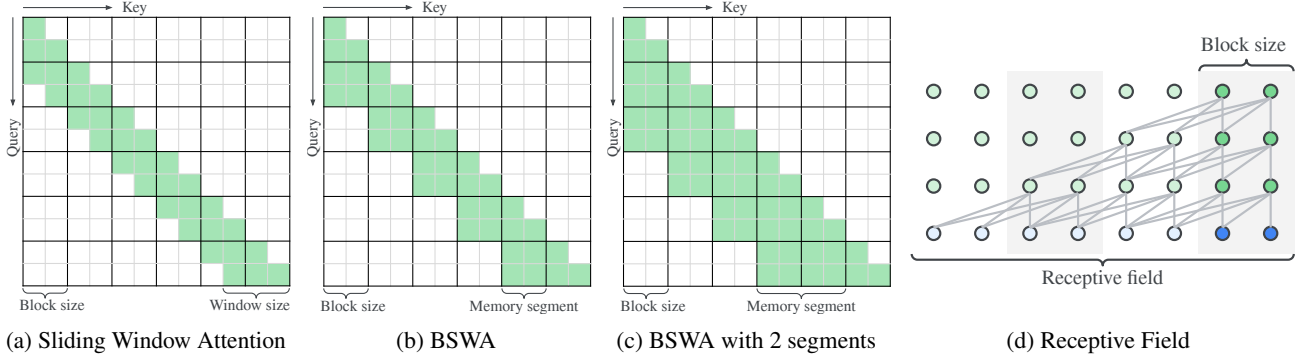
Figure 1: Comparison of query-key attention masks for Sliding Window Attention (SWA) variants. (a) Sliding Window Attention: Attention is restricted to the current window = 3. (b) Block Sliding Window Attention (BSWA) (block size = 2, memory segment = 1): Attention is allowed to previous blocks within the memory segment. (c) BSWA (block size = 2, memory segment = 2): The memory segment is expanded, allowing attention to a larger past context. (d) Illustrates the receptive field of BSWA (block size = 2, memory segment = 1, depth = 4): The region within the curly braces represents the receptive field.
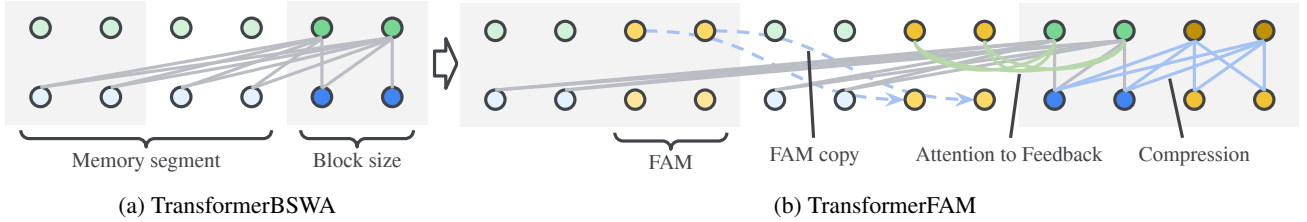

Figure 2: Comparison of attention patterns in Transformer layer. (a) TransformerBSWA: Input query attends to the current block and two memory segments, providing past context. (b) TransformerFAM: Input query attends to the current block, memory segments, and past FAM (green lines). FAM query (copied from previous FAM, blue dash arrow) compresses the current block to update FAM. This feedback loop enables information compression and propagation over indefinite horizon, which is working memory. Figure 5 shows in detail how the dynamic process occurs over time.

the brain areas correlated with the attention overlap substantially with the multisensory cortical regions involved in multisensory integration. Working memory exhibits similar distribution (Christophel et al., 2017), overlapping with attention-correlated areas. This indicates that attention is an important factor in not only multimodal fusion but also working memory.

In the human brain, working memory (Wallace, 1960) provides a temporary memory for performing tasks. While working memory is stored in sustained activations, long-term memory is stored in weights (Fuster, 1973). LLMs has enormous long-term memory enough to store the entire internet (Villalobos et al., 2022), but they do not have working memory. The activation of working memory is sustained by prefrontal cortical-thalamic loops (Ashby et al., 2005), which means working memory is sustained by the continuous spiking of activation within a feedback loop.

Attention has been shown to be effective for processing both homogeneous and heterogeneous data. The next natural step is to apply attention to its own latent representations through a feedback loop. We hypothesize that this next step will naturally lead to the emergence of working memory in Transformers.

Assumption 1.1. The attention mechanism within the feedback loop functions as working memory.

Feedback connections are prevalent in biological neural networks. Even organisms with simple neural structures, such as C. elegans (with only 302 neurons) (White et al., 1986), exhibit various feedback loops, like connections from higher-level interneurons to lower-level ones (Hasani et al., 2018). However, incorporating feedback loops in Transformer is challenging. There are two main approaches. The first approach is linking the topmost layer to the bottommost (Fan et al., 2020; Bulatov et al., 2022). However, this cannot model feedback between interneurons, and this has only one global working memory. The second approach is within-transformer-block feedback: the output activation of a Transformer layer is fed back as input to the same layer. This is the approach we propose, which enables each

Transformer layer to have a distributed working memory that corresponds to its abstraction level.

We propose a novel Transformer architecture (Transformer-FAM) that enables attention to both homogeneous sequence data and latent representations via a feedback loop. This architecture change fosters the natural emergence of working memory within Transformers. During inference, TransformerFAM has a computational complexity of $O(L)$ and a memory complexity of $O(1)$, where L is the length of the processed tokens. TransformerFAM can maintain past information for an indefinite horizon, making it a promising solution for LLMs to handle infinitely long input sequences.

TransformerFAM does not introduce new weights to the Transformer, allowing the reuse of pre-trained checkpoints. Our experiments show that fine-tuning TransformerFAM with LoRA for just 50k steps significantly enhances performance on long-context tasks across 1B, 8B, and 24B Flan-PaLM LLMs (Chung et al., 2022).

2. TransformerFAM

2.1. Block Sliding Window Attention (BSWA)

In general, when handling long context inputs, there are the two main approaches. A first approach is to increase the context length with increasing the computational resources (memory and processing power). A second approach is the implementation of Sliding Window Attention (SWA) (Beltagy et al., 2020; Jiang et al., 2023), as illustrated in Figure 1a. During inference, the standard implementation allocates key and value cache twice the window length at the beginning, and then using a ring buffer to update the necessary components at each step, in order to avoid memory allocation and copying operations every step, which are computationally expensive.

Big Bird (Zaheer et al., 2020) introduced Sliding Window Attention, which caches on a block-by-block basis. We will refer to this as Block Sliding Window Attention (BSWA). BSWA does not mask out past keys and values in the ring buffer, but attends to all the information in the ring buffer.

BSWA has two hyperparameters: block size and memory segment, as illustrated in Figure 1b and 1c. Block size determines how many tokens are in each block and is also used as the stride for sliding. Memory segment determines how many past blocks to cache. In our experiments, we set the default hyperparameters as follows: block size of 1024 and memory segment of 3 (corresponding to a window size ranging from 3073 to 4096).

Given a sequence input $I = [i_1, i_2, \ldots, i_T]$ to a vanilla Transformer layer, Transformer transforms the input to a sequence output $O = [o_1, o_2, \ldots, o_T]$ as follows (Vaswani et al., 2017):

$$Q, K, V = \text{QKV}(\text{PreLN}(I))$$
$$a_t = \text{SelfAttention}(q_t, K, V) + i_t$$
$$O = \text{FF}(\text{PreLN}(A)) + A$$

While PreLN (Xiong et al., 2020) is used in the equation, it is not mandatory. Autoregressive Transformer changes the SelfAttention equation as follows:

$$a_t = \text{SelfAttention}(q_t, K_{:t}, V_{:t}) + i_t$$

In a Transformer with SWA, the equation is modified to account for a window size of w:

$$a_t = \text{SelfAttention}(q_t, K_{t-w:t}, V_{t-w:t}) + i_t$$

In a Transformer with BSWA, τ denotes the block index. Each block τ contains a set of keys and values, determined by the block size. The equation is modified to account for a block index τ and a memory segment m in Equation (1). $K_{\tau-m:\tau-1}$ is the keys in the memory segments from the m blocks before τ to the block before τ, and $K_{\tau,:t}$ is the keys from the beginning of τ block up to t. We will refer to Transformer with BSWA as TransformerBSWA.

$$\hat{K}_t = \text{Concat}(K_{\tau-m:\tau-1}, K_{\tau,:t}) \tag{1a}$$
$$\hat{V}_t = \text{Concat}(V_{\tau-m:\tau-1}, V_{\tau,:t}) \tag{1b}$$
$$a_t = \text{SelfAttention}(q_t, \hat{K}_t, \hat{V}_t) + i_t \tag{1c}$$

Algorithm 1 presents TransformerBSWA (Equation (1)), re-expressed from the perspective of a block index τ. Algorithm 1 describes how to iteratively calculate a_t and then concatenate the results into an A_τ sequence. Typically, standard implementations employ a causal attention mask to enable parallel computation of self-attention.

TransformerXL (Dai et al., 2019) proposed to use a technique called "stop gradient" for the memory segment. However, we argue that this technique has a negative impact on ability of the model to attend to past information. Specifically, we show that using stop gradient results in a much shorter receptive field than the theoretical receptive field in Section 3.2. We believe that allowing gradient to flow to the memory segment is necessary for the model to learn to carry important information in the memory segment.

Algorithm 1 The function of TransformerBSWA

Input: $I_\tau, K_{\tau-m:\tau-1}, V_{\tau-m:\tau-1}$
Output: O_τ
Function Xformer($I_\tau, K_{\tau-m:\tau-1}, V_{\tau-m:\tau-1}$):
$\quad Q_\tau, K_\tau, V_\tau \leftarrow \text{QKV}(\text{PreLN}(I_\tau))$
$\quad \hat{K}_t \leftarrow \text{Concat}(K_{\tau-m:\tau-1}, K_{\tau,:t})$
$\quad \hat{V}_t \leftarrow \text{Concat}(V_{\tau-m:\tau-1}, V_{\tau,:t})$
$\quad a_t \leftarrow \text{SelfAttention}(q_t, \hat{K}_t, \hat{V}_t) + i_t$
$\quad O_\tau \leftarrow \text{FF}(\text{PreLN}(A_\tau)) + A_\tau$
$\quad$ **return** O_τ
End Function

Algorithm 2 The function of TransformerFAM

Input: $I_\tau, K_{\tau-m:\tau-1}, V_{\tau-m:\tau-1}, F_{\tau-1}$
Output: O_τ, F_τ
Function Xformer($I_\tau, K_{\tau-m:\tau-1}, V_{\tau-m:\tau-1}, F_{\tau-1}$):
$\quad Q_\tau, K_\tau, V_\tau \leftarrow \text{QKV}(\text{PreLN}(I_\tau))$
$\quad Q^F_{\tau-1}, K^F_{\tau-1}, V^F_{\tau-1} \leftarrow \text{QKV}(\text{PreLN}(F_{\tau-1}))$
$\quad \hat{K}_t \leftarrow \text{Concat}(K_{\tau-m:\tau-1}, K^F_{\tau-1}, K_{\tau,:t})$
$\quad \hat{V}_t \leftarrow \text{Concat}(V_{\tau-m:\tau-1}, V^F_{\tau-1}, V_{\tau,:t})$
$\quad a_t \leftarrow \text{SelfAttention}(q_t, \hat{K}_t, \hat{V}_t) + i_t$
$\quad O_\tau \leftarrow \text{FF}(\text{PreLN}(A_\tau)) + A_\tau$
$\quad \tilde{K}_\tau \leftarrow \text{Concat}(K^F_{\tau-1}, K_{\tau,:t})$
$\quad \tilde{V}_\tau \leftarrow \text{Concat}(V^F_{\tau-1}, V_{\tau,:t})$
$\quad A^F_\tau \leftarrow \text{SelfAttention}(Q^F_{\tau-1}, \tilde{K}_\tau, \tilde{V}_\tau) + F_{\tau-1}$
$\quad F_\tau \leftarrow \text{FF}(\text{PreLN}(A^F_\tau)) + A^F_\tau$
$\quad$ **return** O_τ, F_τ
End Function

While this modification might seem to burden training memory and computation, it does not significantly impact performance in practice. This is primarily due to the prevalence of gradient checkpointing (Chen et al., 2016) in LLM training on ML accelerators, as memory often presents the primary bottleneck. Since the attention calculation is recomputed during back propagation, the absence of gradient flow through the memory segment yields negligible consequences for overall gradient computations.

In addition, when training an LLM with sliding window attention on long context inputs (more than 8k tokens), computing the attention over the entire input at once would require too much memory. As a result, the standard practice is to divide the attention into blocks and calculate it using a vectorized map (e.g., jax.vmap, torch.vmap). This reduces the peak memory usage to the amount required to calculate one block. Blocks are independent of each other, so they can be calculated in any order. Understanding this point is essential when evaluating the memory requirements and training efficiency of Feedback Attention Memory (FAM), as discussed in Section 2.2.

Transformer has quadratic memory and computation com-

plexity with respect to the length of the input sequence due to self-attention. It has $O(L^2)$ complexity for input length L. However, Transformer with Sliding Window Attention has linear complexity with respect to the input sequence. It has $O(L \times W)$ complexity for input length L and window size W. If the input length is about the same as the window size ($\sim$ 1k), the complexity difference is almost negligible, but if the input length is large like 128k in GPT-4 turbo (Achiam et al., 2023), there is a huge difference. In inference, Transformer with SWA or BSWA only needs to cache a fixed ring buffer (block size + memory segment). Therefore, it only consumes constant memory regardless of the generated token length. Therefore, LLMs using SWA or BSWA can generate infinitely long output tokens.

However, BSWA has a limited receptive field, approximately equal to model depth $\times$ window size as illustrated in Figure 1d. As a result, the later generated tokens are not related to tokens outside the receptive field (e.g., prompt). To address this limitation, we propose a novel architecture in the following Section 2.2. Our approach, Feedback Attention Memory (FAM), builds upon BSWA. This is because the block stride concept of BSWA is well-suited for blockwise feedback updates.

2.2. Feedback Attention Memory

As mentioned in Section 1, we hypothesized that attending to the feedback loop can give rise to working memory in Assumption 1.1. To implement the feedback loop, we add feedback activations that feed contextual representation back into each block of BSWA. We call these virtual activations as Feedback Attention Memory (FAM). FAM is designed to meet the following key requirements:

- Integrated Attention: Self-attention should simultaneously process input context and FAM.

- Block-Wise Updates: FAM should be updated when transitioning between blocks.

- Information Compression: FAM updates should compress current block information, conditioned on previous FAM.

- Global Contextual Storage: FAM should store comprehensive contextual information indefinitely.

The proposed architecture achieves this by appending FAM to block segments and incorporating it into self-attention processes. This enables richer representations and dynamic propagation of global contextual information across blocks, as illustrated in Figure 2b. When self-attention occurs on the current block, the input query for the block attends to the input key for that block, the memory segment, and the previous FAM. The previous FAM provides global contextual

information, allowing for a much richer representation than BSWA. In parallel, the FAM query attends to the current block and the FAM key. The FAM query compresses the current block, conditioned on the previous global contextual information. The FAM query is dynamically generated based on previous global contextual information, as it is copied from the previous FAM. Then, the newly updated FAM serves to propagate global contextual information to the next block recursively. This process is formally described in Algorithm 2.

While Algorithm 2 might initially suggest a doubling of matrix operations compared to Algorithm 1, it performs the same number of matrix operations in the actual implementation, because it starts with the concatenation of block input I_τ and FAM $F_{\tau-1}$. The attention mask within self-attention requires a minor modification to accurately represent FAM. The FAM $F_{\tau-1}$ is much shorter than the input I_τ, and in Section 3, we experimented with a block size of 1024 and a FAM length of 64.

Transformers are much better at exploiting the parallelism of ML accelerators than Recurrent Neural Networks (RNNs). This is because RNNs have a causal relationship between input sequences, while Transformers only have a causal relationship between the inputs and the layer one depth below. It is possible to worry that the feedback mechanism of TransformerFAM will eliminate the advantages of Transformers and make training inefficient. As explained in the implementation of BSWA, memory-efficient implementations perform self-attention in blocks using vectorized maps. Otherwise, peak memory increases during LLM training, requiring more ML accelerators. The causal relationship of TransformerFAM only exists between blocks. Since vectorized maps are used to perform self-attention in blocks, the causal relationship between blocks does not affect training speed and memory. In addition, processing 64 additional FAM when processing 1024 block input sequences has only a minor impact on performance. Therefore, the memory consumption and training speed of TransformerFAM are almost the same as those of TransformerBSWA.

TransformerFAM requires additional considerations for FAM initialization and length extrapolation. These details are explained in Appendix B.

An evaluation of multiple FAM variants was conducted, and the best-performing variant is presented in the main paper. Appendix C provides further details for the remaining variants.

3. Experiments

3.1. Training

Pretraining an LLM from scratch requires a huge amount of resources. TransformerFAM can reuse existing LLM checkpoints because it does not add new weights to the Transformer layer. We reused 1B, 8B, and 24B Flan-PaLM LLMs (Chung et al., 2022) for our experiments. This is a large enough size to prove that TransformerFAM is a general solution for LLMs. The model sizes 1B, 8B, and 24B refer to the size of the plain Transformer, excluding the text embedding table. The models use a 256k sentence piece tokenizer (Kudo & Richardson, 2018), resulting in 400M text embedding table weights for the 1B model and 1B weights for the 8B and 24B models. The detailed model architecture is described in Table 5 in Appendix A.1.

Flan-PaLM is a model that is fine-tuned on top of a pretrained PaLM model (Chowdhery et al., 2023) using instruction finetuning. The instruction data consists of few-shot instructions with 100 to 1k tokens, which are packed into 2.5k tokens for training. This means that individual instruction data are concatenated until they reach 2.5k tokens.

We applied both the TransformerBSWA and TransformerFAM architectures to Flan-PaLM and fine-tuned it for an additional 50k steps. We experimented with different memory segment for both architectures. The block size is set to 1024 and the FAM length is set to 64.

During fine-tuning, we used the same Flan instruction data packed into 8.5k tokens. To maintain a minibatch size of 128 for all models, we used 32 TPUv5 (Jouppi et al., 2023) cores for the 1B model, 64 cores for the 8B model, and 128 cores for the 24B model. If we had used more resources and a larger minibatch size, we might have achieved better results than those reported in the paper.

We performed LoRA finetuning by adding LoRA (Hu et al., 2021) to the Attention and FF layers of Transformer without training all the parameters. Full finetuning resulted in lower scores on various tasks reported by GPT-3 (Brown et al., 2020), because catastrophic forgetting (Kirkpatrick et al., 2017) occurred in domains that were not covered by the instruction data. In LoRA finetuning, the scores on GPT-3 tasks actually improved, and the performance on long context tasks was similar to that of full finetuning. The rank of LoRA was 64, and the weights of the original Attention and FF layers were merged with LoRA weights and used during inference.

The Adafactor optimizer ($\beta_1 = 0.9, \beta_2 = 0.99$) (Shazeer & Stern, 2018) was used with constant learning rate. The learning rates used were 10^{-4} for 1B, and 3×10^{-3} for both 8B and 24B.

In addition, TransformerXL exhibits comparable perfor-

mance to TransformerBSWA. The implementations are almost identical, with TransformerXL employing an additional QK attention mask to mask out keys beyond a predetermined window size. Appendix B.4.5 demonstrates that the performance difference between the two architectures is insignificant, and therefore, experimental results for TransformerBSWA are only included in the main paper.

3.1.1. DATA

The ideal data for training TransformerFAM is a very long document with continuous context, such as textbooks and novels, and the data should be large enough to finetune an LLM. Additionally, the same very long document should be used continuously in each minibatch component, while maintaining FAM and memory segments between training steps.

The loss function of an LLM is to minimize the difference between the parametric probabilistic model $P(x|\theta)$ and the data-generating true distribution $p_{data}(x)$. The Kullback–Leibler divergence (KL divergence) is used to measure this difference. To perform an unbiased estimation of KL divergence, we draw samples from the data-generating true distribution, which are assumed to be independent and identically distributed (IID). However, the ideal training scenario for the aforementioned memory training directly contradicts the IID assumption. We refer to this as the curse of IID.

Due to the curse of IID, we could not find the training infrastructure or data suitable for training memory. So we used Flan instruction data as a last resort.

We used Flan instruction data as training data, packed up to 8.5k tokens. In the Flan paper (Chung et al., 2022), a special attention mask was used to prevent attention between different segments during self-attention, by applying a separate mask to each packed segment. We did not use this special attention mask processing. Attention occurs causally within the window, regardless of the segment. We expected TransformerBSWA and TransformerFAM to learn to remember important information and forget outdated information by themselves.

Each token in Flan data has a weight. Few shots examples and instructions have a weight of 0, and answers have a weight of 1. This means that the model only learns to generate the answer. To incentivize the model to remember long contexts, we randomly selected 256 consecutive tokens from 8.5k tokens and appended them to the end of the data with the prompt '[repeat random segment]:'. The repeated tokens were given a weight of 0.1. We hope that future studies can use more suitable data for training memory, such as long continuous documents, long-form speech, video or video game.

3.2. PassKey Retrieval

The PassKey retrieval task is a recent benchmark used in several long-context transformer papers (Mohtashami & Jaggi, 2023; Tworkowski et al., 2023; Chen et al., 2023). In this task, a passkey is presented at the beginning, followed by a very long filler context. Finally, a question about the passkey is asked, as shown in Figure 11 in Appendix D.1.

This task is a good smoke test to quickly check if information is transmitted in a long context. However, this task only checks if small and important information is transmitted, and does not check if large amounts of information can be efficiently compressed.

We fine-tuned the Flan-PaLM 1B model for 5k steps with the PassKey format, which has a filler context of 2k to 18k randomly. We used a block size of 1024, TransformerBSWA with 0 to 12 memory segments, TransformerFam with 0 memory segments, and a FAM length of 64. When the number of memory segments is 3, the window size is 3k (i.e. memory segment × block size).

As shown in Figure 3, TransformerFAM was able to perfectly solve the task with a filler context of up to 260k tokens. In the figure, MX denotes the number of BSWA memory segments. The performance of TransformerBSWA improves significantly up to M2, after which it saturates. The performance of M12 also drops significantly after 20k tokens. The theoretical receptive field of M2 is 36k (i.e. depth(18) × memory segment(2k)), but the effective receptive field is much shorter.

Figure 3: PassKey Retrieval: Performance across different Transformer models and memory segment configurations. MX denotes the number of BSWA memory segments. FAM represents TransformerFAM with 0 memory segments. TransformerFAM successfully solves the task.

In Figure 3, it is important to compare M1_SG and M1. M1_SG has a stop gradient applied to one memory segment, which limits the receptive field to the window size. This is

because the model cannot learn which contextual information stored in the memory segment will have a good result later. SWA with back propagation through time (BPTT) functions similarly to a time-limited RNN. It is common to use stop gradients on the memory segments of TransformerXL (Dai et al., 2019; Chevalier et al., 2023). However, we recommend against this practice.

In Appendix B.4.6, we compare our work with recent Transformer with memory papers (Bulatov et al., 2022; Chevalier et al., 2023).

3.3. Long Context Tasks

Gemini (Team et al., 2023) evaluated long-context capabilities using the following tasks: NarrativeQA (Kočiský et al., 2018), Scrolls-Qasper, Scrolls-Quality (Shaham et al., 2022), and XLSum (Hasan et al., 2021). Additionally, PG-19 (Rae et al., 2019) and Isabelle (Wu et al., 2022) are another common evaluation tasks among long-context Transformer papers (Rae et al., 2019; Wu et al., 2022; Chen et al., 2023). Detailed information on the evaluation data is provided in Table 10 in Appendix D.2.
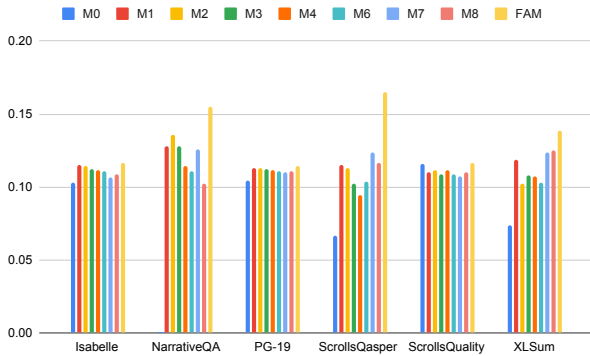

Figure 4: Normalized scores of long-context tasks evaluated by Flan 1B with different Transformer models and different memory segment configurations. FAM outperforms all other BSWA configurations.

We evaluated the long-context capabilities of the 1B TransformerBSWA model trained in Section 3.1 using memory segment sizes ranging from 0 to 8. As shown in Figure 4, TransformerFAM outperformed TransformerBSWA on all the long context tasks (LCT), regardless of the number of memory segments in BSWA. It shows a significant performance improvement on ScrollsQasper and NarrativeQA, where it has to understand 5k to 500k tokens of context before answering a question. The LCT results demonstrate that TransformerFAM can effectively compress and retain important contextual information within extremely long contexts.

Above M1, the number of memory segments does not significantly impact LCT performance on TransformerBSWA, because the input sequences are much longer than the window size of all experiments. We observed the same phenomenon in TransformerFAM, and TransformerFAM uses 3 memory segments in Figure 4. The figure shows the normalized scores of all tasks to view the scores on the same scale. The raw results are in Table 11 in Appendix D.2.

We further evaluated TransformerFAM and TransformerB-SWA on 8B and 24B models. As shown in Table 1, TransformerFAM demonstrates scalability as the model size increases. This suggests that self-attention can route local information relevant to each input sequence while simultaneously routing contextual information to the FAM.

Model	BSWA 8B	FAM 8B	BSWA 24B	FAM 24B
Isabelle	82.1	**82.5**	86.6	86.6
NarrativeQA	18.4	**19.3**	22.6	**23.0**
PG-19	52.4	**52.9**	55.7	**57.2**
ScrollsQasper	12.4	**18.5**	28.0	**29.4**
ScrollsQuality	47.3	**48.5**	55.4	**58.0**
XLSum	22.0	**24.7**	24.7	**26.4**

Table 1: LCT scores on 8B and 24B models comparing TransformerBSWA and TransformerFAM

In addition, TransformerFAM marginally surpasses TransformerBSWA on GPT-3 tasks (Brown et al., 2020) (see Table 2). This is not expected because all tasks are shorter than 2k tokens. We hypothesize this improvement stems from the efficient contextual representation by TransformerFAM. By offloading contextual data to FAM, TransformerFAM reduces duplication within input activations, optimizing latent space usage.

Model	GPT-3 Rank	GPT-3 Gen
BSWA 1B	60.2	33.9
FAM 1B	**61.0**	**34.7**
BSWA 8B	72.8	54.3
FAM 8B	**74.0**	**54.9**
BSWA 24B	78.2	62.6
FAM 24B	**78.5**	**63.4**

Table 2: GPT-3 Task Report: Summarizes GPT-3 performance on ranking and generative tasks. (Details in Table 12)

Thus, BSWA memory segments (local representation) and FAM (global representation) complement each other. For LLMs, we recommend using FAM for compressed contextual representation alongside BSWA memory segments up to inference budgets (e.g., 2k, 8k, 32k (Team et al., 2023), or 128k (Achiam et al., 2023)).

3.4. Ablation study

3.4.1. FAM LENGTH

In the Flan 1B model, we observed performance saturation on ScrollsQasper, ScrollsQuality, and XLSum tasks when the FAM length reached 64. Interestingly, performance declined when FAM length exceeded 64. This suggests that information compression is more effective when space is limited. Miller's Law (Miller, 1956) states that the average person can only hold about 7 (plus or minus 2) items in their working memory at a time. The limited space is a feature, not a bug.

FAM Len.	ScrollsQasper	ScrollsQuality	XLSum
4	5.0	27.1	15.1
16	6.0	25.2	15.2
64	**7.2**	**27.9**	15.9
256	5.1	26.5	**16.0**
1024	5.3	26.3	**16.0**

Table 3: LCT scores according to FAM Length

3.4.2. THE NUMBER OF PREVIOUS FAM BLOCKS

In Figure 2b, the input query attends to the FAM as denoted Attention to Feedback. The input query can attend to not only the immediately previous FAM, but also to more previous FAMs. Table 4 shows the XLSum scores for different numbers of previous FAM blocks. As the table shows, increasing the number of blocks did not have a significant effect, because the previous FAM already encapsulates all the previous information by a feedback copy. Therefore, the default setup attends to only the immediately previous FAM.

FAM blocks	1	2	3	4	6
XLSum	15.9	15.8	15.7	15.1	16.2

Table 4: XLSum scores according to the number of FAM blocks

In Appendix B.4, additional ablation studies are presented, due to the page limitation.

4. Related Work

There have been attempts to incorporate feedback mechanisms into the Transformer, but most of them involve feeding the output activations from the top layer to the bottom (Bulatov et al., 2022; Chevalier et al., 2023) or to intermediate layers (Fan et al., 2020). Since the top three layers in the Transformer are heavily focused on output reconstruction (Pasad et al., 2021), we hypothesize that there is a significant representational gap between the top and other layers. In this paper, we propose a feedback mechanism between intermediate layers.

There were papers that compressed information block-wise (Rae et al., 2019; Guo et al., 2019; Gupta & Berant, 2020; Mohtashami & Jaggi, 2023; Mu et al., 2023). However, in those papers, the information was not propagated infinitely. Relevant prior work includes the use of recurrent cross-attention between blocks (Hutchins et al., 2022), enabling the propagation of compressed information to subsequent blocks. Additionally, incorporating feedback from a few upper layers has been used to integrate past information (Ju et al., 2022). We propose TransformerFAM under the assumption that the human brain processes homogenous, heterogeneous, and feedback data with the same attention mechanism across distributed brain areas.

In Appendix E, additional related works are presented, due to the page limitation.

5. Conclusion

In the film 'Memento' (2000), the protagonist struggles with anterograde amnesia, which means he can not remember anything before happened in the last 10 minutes, but his long-term memory is intact, He has to tattoo important information on his body to remember it. This is similar to the current state of large language models (LLMs). LLMs memorize the entire internet thanks to scaling laws (Kaplan et al., 2020), which allow them to store an enormous amount of information in large weights (long-term memory). However, their short-term memory is limited by the attention window. As a result, the complex prompt engineering becomes necessary to help them recall important details. We propose a new architecture called TransformerFAM that could fix anterograde amnesia of LLMs.

The rapid progress of machine learning is astonishing, but there are two key problems that we still do not know how to approach: reasoning and memory. In this paper, we provide a clue to the memory problem. Memory is a critical prerequisite for reasoning. It is hard to imagine how we can derive complex mathematical equations without working memory. Reasoning must be a phenomenon that occurs based on the current working memory.

This paper explores the integration of attention-based working memory, a concept from neuroscience, into the field of deep learning. Our goal is to ignite further research within the community to address and solve the ongoing challenge of limited memory in deep learning. There is a significant set of problems to tackle here, ranging from refining feedback attention architecture to investigating the transfer of working memory to long-term memory.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Ashby, F. G., Ell, S. W., Valentin, V. V., and Casale, M. B. Frost: A distributed neurocomputational model of working memory maintenance. *Journal of cognitive neuroscience*, 17(11):1728–1743, 2005.

Baars, B. J. Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. *Progress in brain research*, 150:45–53, 2005.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Bulatov, A., Kuratov, Y., and Burtsev, M. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Chevalier, A., Wettig, A., Ajith, A., and Chen, D. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*, 2023.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Christophel, T. B., Klink, P. C., Spitzer, B., Roelfsema, P. R., and Haynes, J.-D. The distributed nature of working memory. *Trends in cognitive sciences*, 21(2):111–124, 2017.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Fan, A., Lavril, T., Grave, E., Joulin, A., and Sukhbaatar, S. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*, 2020.

Fuster, J. M. Unit activity in prefrontal cortex during delayed-response performance: neuronal correlates of transient memory. *Journal of neurophysiology*, 36(1): 61–78, 1973.

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

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X., and Zhang, Z. Star-transformer. *arXiv preprint arXiv:1902.09113*, 2019.

Gupta, A. and Berant, J. Gmat: Global memory augmentation for transformers. *arXiv preprint arXiv:2006.03274*, 2020.

Hasan, T., Bhattacharjee, A., Islam, M. S., Samin, K., Li, Y.-F., Kang, Y.-B., Rahman, M. S., and Shahriyar, R. Xlsum: Large-scale multilingual abstractive summarization for 44 languages. *arXiv preprint arXiv:2106.13822*, 2021.

Hasani, R., Lechner, M., Amini, A., Rus, D., and Grosu, R. Can a compact neuronal circuit policy be repurposed to learn simple robotic control? *arXiv preprint arXiv:1809.04423*, 2018.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Hutchins, D., Schlag, I., Wu, Y., Dyer, E., and Neyshabur, B. Block-recurrent transformers. *Advances in Neural Information Processing Systems*, 35:33248–33261, 2022.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Jouppi, N., Kurian, G., Li, S., Ma, P., Nagarajan, R., Nai, L., Patil, N., Subramanian, S., Swing, A., Towles, B., et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1–14, 2023.

Ju, D., Roller, S., Sukhbaatar, S., and Weston, J. E. Staircase attention for recurrent processing of sequences. *Advances in Neural Information Processing Systems*, 35:13203–13213, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Kudo, T. and Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

LMSYS. Lmsys chatbot arena leaderboard. https://chat.lmsys.org/?arena, 2023.

Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

Mohtashami, A. and Jaggi, M. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

Mu, J., Li, X. L., and Goodman, N. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2023.

Narayanan, A., Prabhavalkar, R., Chiu, C.-C., Rybach, D., Sainath, T. N., and Strohman, T. Recognizing long-form speech using streaming end-to-end models. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pp. 920–927. IEEE, 2019.

Oren, M., Hassid, M., Adi, Y., and Schwartz, R. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*, 2024.

Pasad, A., Chou, J.-C., and Livescu, K. Layer-wise analysis of a self-supervised speech representation model. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 914–921. IEEE, 2021.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K. K., et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., et al. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*, 2022.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Tang, X., Wu, J., and Shen, Y. The interactions of multisensory integration with endogenous and exogenous attention. *Neuroscience & Biobehavioral Reviews*, 61: 208–224, 2016.

Tay, Y., Dehghani, M., Abnar, S., Chung, H. W., Fedus, W., Rao, J., Narang, S., Tran, V. Q., Yogatama, D., and Metzler, D. Scaling laws vs model architectures: How does inductive bias influence scaling? *arXiv preprint arXiv:2207.10551*, 2022.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Tworkowski, S., Staniszewski, K., Pacek, M., Wu, Y., Michalewski, H., and Miłoś, P. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*, 2023.

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

Villalobos, P., Sevilla, J., Heim, L., Besiroglu, T., Hobbhahn, M., and Ho, A. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*, 2022.

Wallace, A. F. Plans and the structure of behavior, 1960.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

White, J. G., Southgate, E., and Thomson, J. N. S. brenner (1986) the structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 314:1–340, 1986.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138–14148, 2021.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

A. Architecture details

A.1. Flan-PaLM architecture

Table 5 provides detailed information on the architecture of Flan-PaLM for 1B, 8B, and 24B models. MQA (Multi-Query Attention) (Shazeer, 2019) is an attention mechanism that employs a single set of keys and values for all attention heads.

Component	Flan-PaLM 1B	Flan-PaLM 8B	Flan-PaLM 24B
Num. Layers	18	32	56
Model Dim.	1536	4096	4096
FF Multiplier	8	4	8
Num. Heads	12	32	32
Use MQA	F	F	T

Table 5: Architecture of 1B, 8B, and 24B Flan-PaLM models

A.2. FAM hyperparameters

Table 6 presents the default settings for the hyperparameters added in TransformerFAM.

Component	Value
Memory Segment	3
FAM length	64
Probability of Random State Passing	0.8

Table 6: TransformerFAM hyperparameters

B. Additional details of TransformerFAM

The appendix describes additional details not covered in the main text.

B.1. Initial FAM

The current FAM F_τ is the output activation of the previous FAM update. So, how do we initialize the FAM for the first block?

When FAM are used as queries, they summarize the block context. Therefore, we learn by adding learnable summarization embeddings to the token embedding lookup level of the Transformer model. This is the same as prepending learnable embeddings in soft prompt tuning (Lester et al., 2021). The difference is that full attention is applied between the FAM prompt activations, and the updated FAM is used for the next block. The FAM prompt is passed to the next transformer layer through the forward propagation of the Transformer model, and it has a summary representation that is suitable for the level of the layer.

Prefix tuning (Li & Liang, 2021) can also be used to train learnable initial FAM at each Transformer layer. However, we had difficulty matching the magnitude of the learnable prefix to the regular input sequence, and the results of prompt tuning were consistently better. Ablation study in Appendix B.4.4 shows prompt tuning outperforms. In addition, prefix tuning has the disadvantage of adding additional weights of FAM length to each layer.

In addition, the first FAM update in self-attention should utilize a zero tensor in the residual connection rather than the initial FAM, because the initial FAM does not carry any contextual information.

In summary, we learned the initial FAM using prompt tuning, which only adds a very small number of weights of FAM length to the entire model. We share the same initial FAM across all minibatches.

B.2. Input Length Extrapolation

B.2.1. FAM POSITION ENCODING

We used rotary position embedding (RoPE) (Su et al., 2024) in all of our experiments. Each input sequence is assigned an integer position m, which is converted into sinusoidal encoding in the form of $\exp(im\theta)$ where $\theta = 10000^{-2i/d_{model}}$ (Vaswani et al., 2017).

FAM is inserted at each block boundary, but the problem is how to assign positions to FAM. We assigned positions to FAM in order from the last number of the compressed block. For example, if the block size is 4, the FAM length is 2, and the positions of the compressed blocks are m, $m + 1$, $m + 2$, and $m + 3$, then the updated FAM positions are $m + 2$ and $m + 3$.

We tried other methods, but this method worked best according to the ablation results in Appendix B.4.1.

B.2.2. RANDOM POSITION OFFSET

The input length extrapolation problem of Transformer is well known (Chen et al., 2023; Xiong et al., 2023). For example, a Transformer LLM trained with 2k tokens experiences a severe performance drop when generating 8k tokens. This is because machine learning (ML) does not generalize well to situations that it has not seen during training.

Transformer with SWA does not suffer from the position extrapolation problem when using relative positional embedding like RoPE. This is because the score value of $q_m \cdot k_n$ becomes a function of $(m - n)$ in the form of $\exp(i(m - n)\theta)$.

Because the range of $(m - n)$ is limited to the window size, independent of the input length, the model can handle long input sequences without facing novel scenarios during inference. The model can accurately determine the relative position, if the window size is smaller than the maximum wavelength of θ. If the typical θ is used for positional embedding, the working maximum window size is the maximum wavelength ($= 2 \times 10000 \times \pi \sim 63$k tokens).

However, FAM breaks the symmetry of relative position. Since the absolute position from the past to the present is recursively embedded in the FAM, the large absolute position value that the model encounters for the first time during inference creates a situation where the model needs to extrapolate.

We propose Random Position Offset as a solution. At each training step, the Transformer model randomly samples a scalar value between 0 and the maximum wavelength. All Transformer layers add that random value to the absolute position at that training step. Therefore, the FAM experiences the entire range of absolute position embedding during training.

This is a purely training technique. During inference, the default offset is 0. We used the below algorithm that generates 0 by 50% when sampling the offset, as 0 is the default value.

```
offset = np.uniform([b], maxval=wavelen)
offset *= np.round(np.uniform([b]))
```

B.2.3. RANDOM STATE PASSING

Due to the recursive use of FAM, we need to determine the maximum number of updates for which the FAM remains valid. If it is updated up to 8 times during training, the model will have to extrapolate the situation where it is updated 100 times during inference.

The same problem existed in RNNs, and Random State Passing (RSP) (Narayanan et al., 2019) was proposed to handle long-form speech in the speech domain. We also used RSP to generalize the number of updates of FAM.

RSP saves FAM as weights at the end of each training step. Then, it loads the saved FAM at the next training step. When initializing FAM, it either uses randomly saved FAM or learned FAM. In our default setup, it used saved FAM with 80% probability. To save FAM of all minibatch, weights are required as many as the number of minibatch. We save FAM of only the first batch and all minibatch share it in the next training step.

On the other hand, saved FAM can be thought of as learnable prefix for prefix tuning (Li & Liang, 2021). It is also possible to train only the FAM while the model is frozen and use them for various downstream tasks or personalization. This part is left as a future research topic.

B.3. FAM illustrated

In Section 2.2, we define the feedback loop as feedback activations that feed contextual representation back into each block of BSWA. This feedback loop is formally described in Algorithm 2. While Figure 2b illustrates Algorithm 2, the static image makes it challenging to fully grasp the dynamic nature of the decoding self-attention mechanism. To clarify this, we create a multi-frame animation in Figure 5 that demonstrates how the attention mechanism evolves over time (top to bottom).
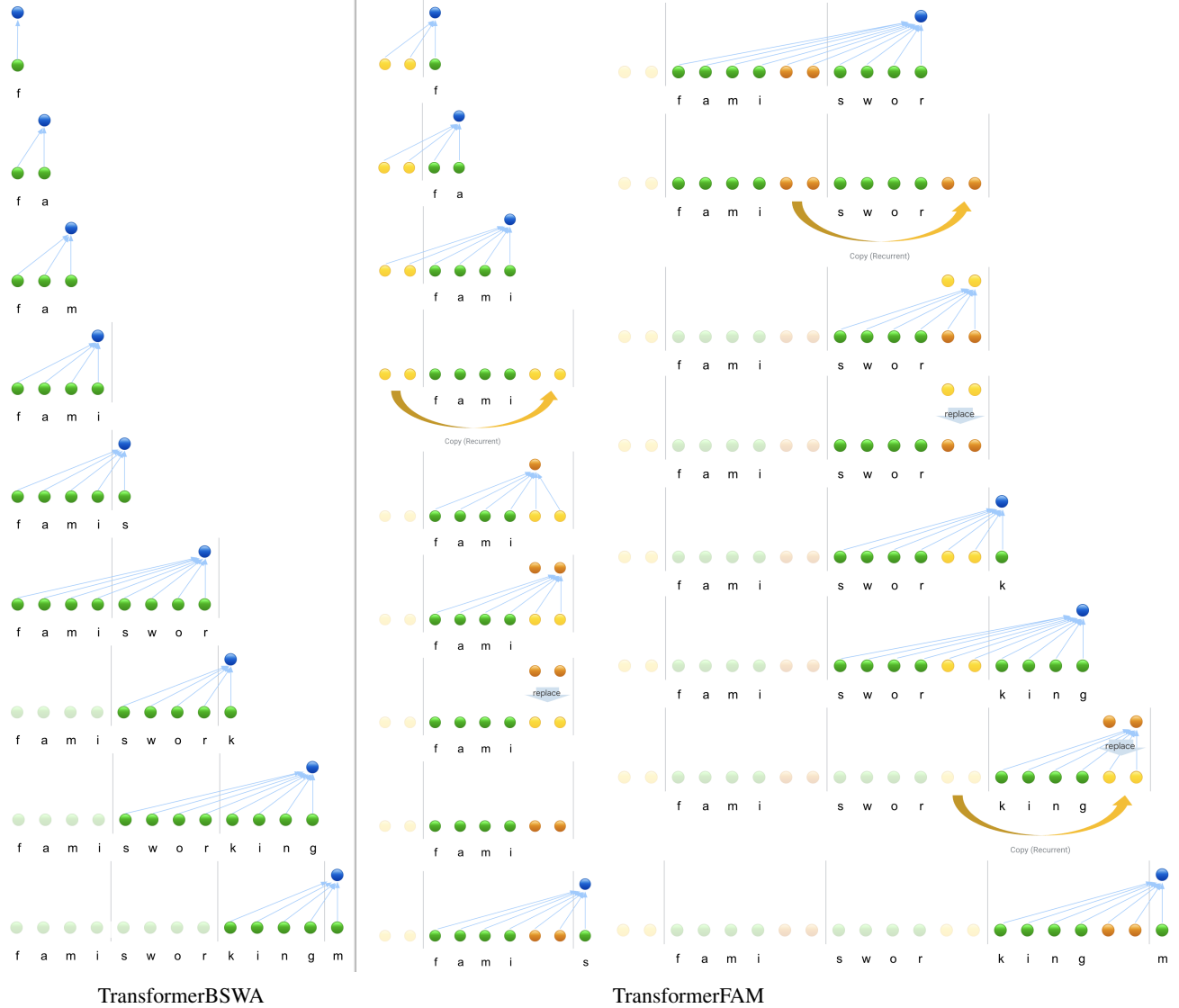

Figure 5: Visualization of self-attention during inference over time. (A) Self-attention pattern of TransformerBSWA layer with memory segment size of 1. (B) Self-attention pattern with FAM added.

B.4. Ablation Studies

All ablation studies were conducted on a 1B model.

B.4.1. FAM POSITION ENCODING

Appendix B.2.1 proposed assigning the last number of the compressed block as the FAM position. In addition, we also experimented with FAM having a float position between blocks. In the example of block size 4 in Appendix B.2.1, the FAM positions would be $m + 3 + 0.33$ and $m + 3 + 0.66$. We also experimented with the case where the FAM position is always

0.

As shown in Table 7, the last number showed the best accuracy.

FAM position	PG-19	Isabelle
Last number	**47.7**	**73.6**
Float number	47.6	73.4
Zero	47.3	72.1

Table 7: PG-19 and Isabelle Accuracy across various FAM Position Encoding

Figure 6 shows FAM (i.e. Last number in Table 7) outperforming FAM-POS (i.e. Zero) in PG-19 accuracy over most base frequencies.

B.4.2. RANDOM POSITION OFFSET

As mentioned in Section B.2.2, the input length extrapolation problem of the Transformer is well-known. The "Attention is All You Need" (Vaswani et al., 2017) introduced sinusoidal position encoding in the form of $\exp(i(m-n)\theta)$ where $\theta = 10000^{-2i/d_{model}}$. Popular solutions for full attention models include increasing the base frequency from 10k to 500k (Chen et al., 2023) or scaling down $(m-n)$ to $(m-n) \times (1024/4096)$ (Xiong et al., 2023).

Since the relative position of BSWA has a range of 0 to window size, it does not suffer from the input length extrapolation problem. However, TransformerFAM encodes absolute position into FAM, which requires a solution. Section B.2.2 proposes Random Position Offset (RPO) as a solution.

In Figure 6, FAM shows better PG-19 accuracy than FAM-RPO at the 10k base frequency. As mentioned in Appendix D.2, the max length of PG-19 was truncated to 256k in the experiments.
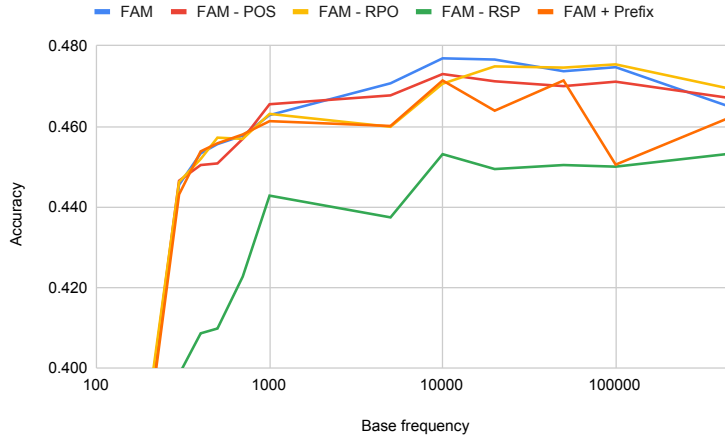

Figure 6: PG-19 accuracy for various ablation studies such as RPO (Random Position Offset), RSP (Random State Passing) and Prefix FAM tuning over different base frequency.

Furthermore, scaling up the base frequency or scaling down $(m-n)$ is only a remedy for pain, not a final solution. It reduces the resolution of the position, which negatively affects the overall Transformer performance, as shown in the figure. Interestingly, the originally proposed 10k is a very good value even in long contexts, until the window size reaches its wavelength (63k).

In addition, we did not observe the attention sink phenomenon (Xiao et al., 2023) in our TransformerBSWA experiments. The paper proposes that Transformers with sliding window attention should store initial tokens in a KV cache for long-context inference. However, the relative position of SWA is restricted to a range of 0 to window size, independent of input sequence

length. Our TransformerBSWA implementation, trained on 8.5k tokens, successfully operated for inference up to 256k tokens without any issues.

B.4.3. RANDOM STATE PASSING

To extrapolate the number of FAM updates, Appendix B.2.3 proposes Random State Passing (RSP). In Figure 6, FAM shows significantly better PG-19 accuracy than FAM-RSP. This demonstrates that RSP plays a crucial role in training FAM.

Figure 7 shows the best performance at a probability of 0.8. At 0.8, the half-life is 3 training steps ($0.512 = 0.8^3$). This means that every 26k (8.5k x 3) tokens, FAM restarts from the beginning with a 50% probability. In other words, if FAM experiences 25 FAM updates during training, it can extrapolate to 256 FAM updates (256k tokens) during inference.
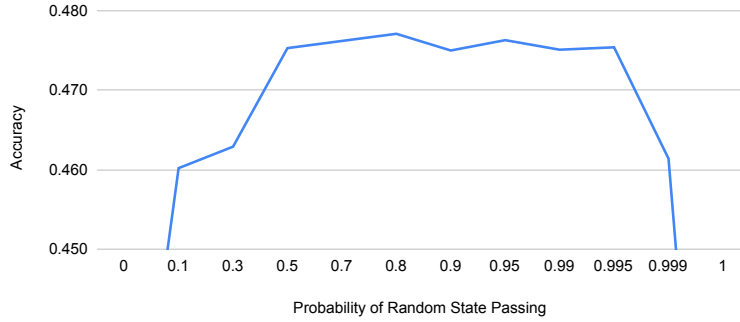

Figure 7: PG-19 accuracy for different probability of Random State Passing.

B.4.4. PROMPT VS. PREFIX

Appendix B.1 mentions that Prompt tuning outperforms Prefix tuning for training FAM in terms of performance, memory, and number of parameters. Figure 6 shows that Prefix tuning for FAM training leads to performance degradation in PG-19 (FAM vs. FAM+Prefix).

B.4.5. TRANSFORMERXL VS. TRANSFORMERBSWA

TransformerXL is a modification of TransformerBSWA that incorporates an additional QK attention mask to mask out keys beyond a specified window size. Therefore, there is no reason to expect a significant difference in performance between the two. Table 8 shows the scores for the main tasks. TransformerBSWA used a single memory segment, and TransformerXL window size was equal to the block size of 1024.

Tasks	TransformerBSWA	TransformerXL
Isabelle	72.6	72.2
NarrativeQA (RougeL)	11.1	11.0
PG-19	46.4	46.1
ScrollsQasper (RougeL)	5.0	4.9
ScrollsQuality (accuracy)	26.3	26.5
XLSum (RougeL)	13.6	13.7
GPT-3 Rank	60.2	59.9
GPT-3 Gen	33.9	33.9

Table 8: Comparing TransformerBSWA and TransformerXL on major tasks

B.4.6. COMPARISON WITH OTHER METHODS

We compared FAM with Recurrent Memory Transformer (RMT) (Bulatov et al., 2022), because RMT also implements memory in Transformer using feedback mechanism from top layer to bottom layer. As shown in Figure 8, RMT showed worse performance than Block Sliding Window Attention (BSWA) with 1 memory segment, in the PassKey retrieval task. FAM solved the PassKey retrieval task, but RMT did not work at all with very long filler token lengths.

RMT is implemented by feeding the output memory of the previous segment as the input memory to the next segment. In the constraint that the input is text embedding and the output is text reconstruction in LLM, RMT has an additional constraint that the latent space of the output memory and the input memory must match. In this situation, RMT fails to remember the PassKey for a very long context. On the other hand, FAM seems to compress, store, and propagate information more effectively by learning memory representation that matches the abstraction level of each layer through training.

AutoCompressors (Chevalier et al., 2023) is an extension of RMT that continuously accumulates blockwise memory of RMT. AutoCompressors theoretically should be able to solve the PassKey task since it maintains all the memory tokens for all blocks. However, as shown in Figure 8, its performance drops sharply after 18k tokens. This is because the model only saw up to 18k tokens during training. It fails to generalize to longer filler token lengths. The AutoCompressors in Figure 8 accumulates 260 memories to support up to 260k tokens.
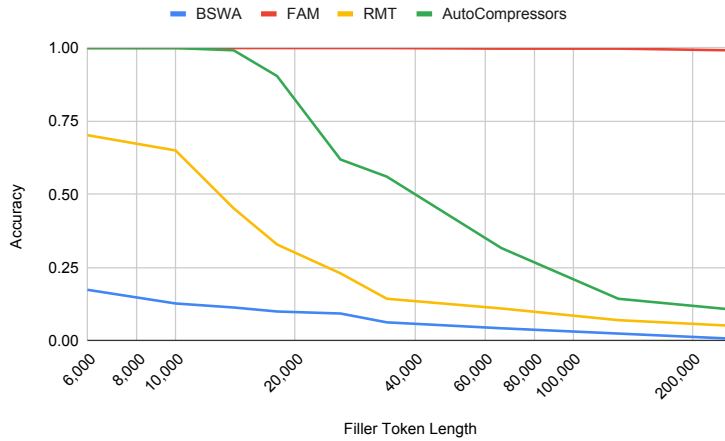

Figure 8: PassKey Accuracy: FAM maintains performance with long sequences, outperforming BSWA, RMT and Auto-Compressors.

C. Don't

There could be various ways to create feedback loops in Transformers. This appendix summarizes our attempts that did not work well. We hope that this will save other researchers time when improving the architecture of feedback loops in future studies.

C.1. Beyond Block-Recurrent Transformers

Block-Recurrent Transformers (BRT) (Hutchins et al., 2022) have a recurrent state that connects each block, and the recurrent state and the block input sequence exchange information through cross-attention. In this architecture, the recurrent state plays a role similar to FAM. We started the project by removing the additional QKV projection and cross-attention for the recurrent state in BRT and integrating all computations into self-attention.

Like BRT, we tried to compress the block input by attention to use it as the recurrent state of the next block, and it required additional projections like BRT. It was difficult to properly train the additional projections that were only used when striding blocks, and as a result, the performance of general tasks such as GPT-3 tasks was degraded. Finally, we found that the activations compressed by attention must go through the FF layer of the Transformer to be aligned as a latent representation

that can be used again as the input of the next Transformer. The input of the Transformer and the output of self-attention are very different representations. However, the FF layer transforms it back into a representation similar to the input of the Transformer for the next Transformer layer. After the discovery of reusing the FF layer, we also found that separate QKV projection and additional projections are not required for FAM. That is how the TransformerFAM architecture was created.

C.2. Feedback Memory Segment

As shown in Figure 2b, TransformerFAM utilizes the updated FAM from the previous block as the input to the current block. Therefore, it is natural to consider using the Transformer outputs of the previous block directly as the input to the current block, instead of using a complex FAM mechanism. Figure 9 illustrates this modification, which we refer to as the Feedback Memory Segment (FM). FM achieves an infinite theoretical receptive field by performing self-attention at the output level. This is a specific variant of Staircase Attention (Ju et al., 2022). Staircase Attention proposes using activations from progressively higher Transformer layers as the memory segment goes further into the past.
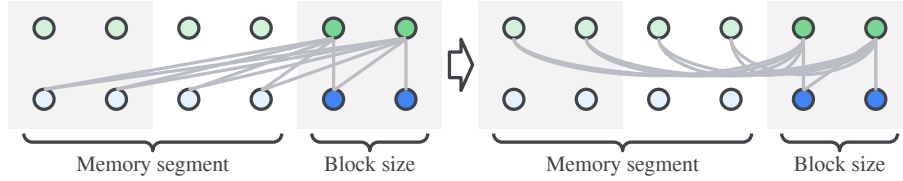

Figure 9: Convert TransformerBSWA to TransformerFM

However, as shown in Figure 10, FM fails to retain PassKey information over a long context. The figure compares M1, 2, and 4 with FM1, 2, and 4 when the memory segment size is 1, 2, and 4 for BSWA and Feedback Memory Segment. TransformerFM outperforms TransformerBSWA, but still falls short of TransformerFAM by a significant margin. In TransformerFM, each activation must possess both local and global representation, similar to TransformerBSWA. However, the absence of an activation specifically responsible for the global representation appears to prevent the retention of critical information like PassKey over an extended context.

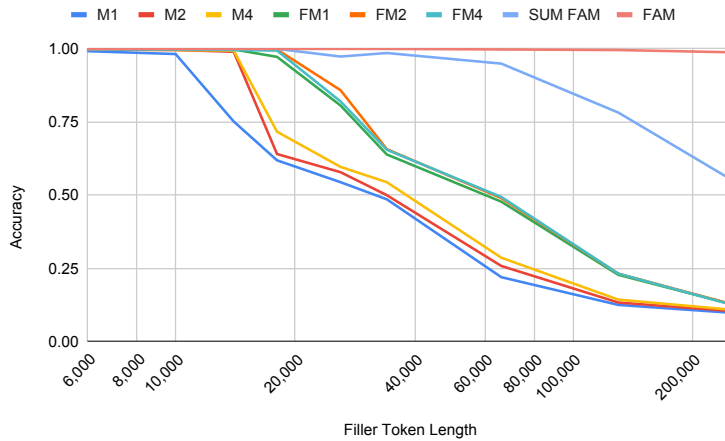Figure 10: PassKey Retrieval Accuracy Using BSWA (M1,2,4), Feedback Memory Segments (FM1,2,4), Static Summary Tokens (SUM FAM) and FAM

Furthermore, TransformerFM exhibited lower performance than TransformerBSWA on GPT-3 tasks (Table 9). As TransformerFM introduced additional complexity without any discernible performance advantages, we discontinued this architecture. Table 9 presents the results after 25k steps of fine-tuning.

Task	Metric	M1	M2	M4	FM1	FM2	FM4
GPT-3 Gen score	EM	32.5	**33.6**	32.9	31.9	32.0	29.7
GPT-3 Rank	Rank Acc.	58.5	**59.0**	58.5	58.2	59.0	58.3
Isabelle	Accuracy	72.5	72.2	70.9	72.6	**73.5**	72.3
PG-19	Accuracy	46.4	46.1	45.9	45.8	**47.2**	44.5
XLSum	ROUGE-L	13.3	**13.9**	**13.9**	13.6	13.6	13.8

Table 9: Comparison of TransformerBSWA and TransformerFM on major tasks

C.3. Static Summary Tokens

As illustrated in Figure 2b, "FAM copy" duplicates the previously updated FAM to the current FAM input. However, "FAM copy" is not an essential component in the feedback loop. When FAM compresses the current block, it can also attend to the previous FAM as both key and value, as shown in the experiment in Section 3.4.2. Therefore, "FAM copy" is not strictly necessary because past FAM information is propagated through self-attention.

As mentioned in Appendix B.1, we learn the initial FAM using prompt tuning, and this token can be considered a "Summary" token. An alternative design could employ a persistent summary token for block compression, with the feedback loop enabled by attending to the key-value pairs of previous FAMs.

As illustrated in Figure 10, SUM FAM (static summary token) prevents the model from successfully solving the PassKey task. This suggests that the query for summarization needs to be dynamically generated conditioned on the past FAM. A static query may not be able to cover all situations effectively. Additionally, simply attending to the key and value alone does not transmit sufficient information from the past. On the other hand, the "FAM copy" mechanism propagates information from the past FAM to the current FAM through the residual connection of the Transformer, which facilitates better performance on the PassKey task.

C.4. Diversity Loss

In Section 3.4.1, we observed a performance drop when the FAM length exceeded 64. We hypothesized that FAM was underutilizing its capacity and experiencing mode collapse, where lengthy FAM either focused on a small number of inputs or prevented all inputs from obtaining sufficient information.

As a remedy, we employed diversity loss as an auxiliary loss, similar to Wav2vec 2.0 (Baevski et al., 2020). Diversity loss aims to maximize the entropy of QK attention probability, as shown in Equation (2b). This loss encourages FAM to uniformly attend to all inputs, and all inputs equally attend to FAM. In Equation (2), b denotes the batch size, τ represents the block index, l indicates the sequence position within a block, and h refers to multi heads. In Equation (2a), $\bar{p}_{b\tau}$ represents the average attention probability across all sequences and multi-heads for the self-attention of each block. The auxiliary loss functions to regularize this probability towards uniformity.

$$\bar{p}_{b\tau} = \frac{1}{HL} \sum_{h=1}^{H} \sum_{l=1}^{L} p_{b\tau hl} \tag{2a}$$

$$\mathcal{L}_d = \frac{1}{BT} \sum_{b=1}^{B} \sum_{\tau=1}^{T} -H(\bar{p}_{b\tau}) = \frac{1}{BT} \sum_{b=1}^{B} \sum_{\tau=1}^{T} \bar{p}_{b\tau} \log \bar{p}_{b\tau} \tag{2b}$$

We trained the model with diversity loss with various weights, but the overall performance was always worse, regardless of the FAM length. It did not help at all even when the FAM length was 256 or longer.

C.5. Reconstruction Loss

Compressive transformers (Rae et al., 2019) compress memory segments before forwarding them to the next block. The key difference between our approach is that the compressed information is not recurrently connected. The paper proposes a reconstruction loss as an auxiliary loss, which aims to reconstruct the original activations from the compressed activations.

We also experimented with the reconstruction loss. In details, we generated transformer outputs autoregressively from the updated FAM and the query sequence of the original block, and compared them to the original outputs using the MSE loss. However, this did not help to improve the performance.

D. Experiments details

This section provides additional details about the experiments that were not covered in Section 3.

D.1. PassKey Retrieval

The format of the PassKey retrieval task is shown in Figure 11. The original paper (Mohtashami & Jaggi, 2023) also included a prefix filler, but we removed it in our paper.

```
There is an important info hidden inside a lot of irrelevant text.  Find it and
memorize them.  I will quiz you about the important information there.
The pass key is <PASS KEY>.  Remember it.  <PASS KEY> is the pass key.
<filler>
What is the pass key?  The pass key is <PASS KEY>
```

Figure 11: PassKey Retrieval Format (Mohtashami & Jaggi, 2023)

D.2. Long Context Tasks

Table 10 provides detailed information about the long context tasks we used. Due to the limitations of TPU memory, PG-19, Isabelle, and NarrativeQA were truncated to 256k tokens, which is within two standard deviations.

Eval task	Metric	Num.	Max len.	Mean len.	Standard deviation	Description
Isabelle	Next Token Accuracy	16	280500	60874	65008	Formal theorems
NarrativeQA	ROUGE-L	5878	505949	85449	84376	QA on a given underlying narrative
PG-19	Next Token Accuracy	50	491500	94704	82677	Project Gutenberg books published before 1919
ScrollsQasper	ROUGE-L	1726	24223	5027	2580	QA on scientific research papers
ScrollsQuality	Accuracy	8344	9294	6220	2050	QA on a given story or document
XLSum	ROUGE-L	37747	13571	1888	1304	Multilingual abstractive summarization

Table 10: Overview of long-context tasks, including their token count, evaluation metric, and a brief description. The token count is counted after a 256k sentencepiece tokenizer.

Table 11 presents the LCT results for TransformerBSWA and TransformerFAM. MX represents the number of memory segments for TransformerBSWA.

Task	Metric	M0	M1	M2	M3	M4	M6	M7	M8	FAM
Isabelle	Accuracy	65.0	72.6	72.0	70.6	70.3	69.7	67.29	68.4	73.6
NarrativeQA	ROUGE-L	0.0	11.1	11.8	11.2	10.0	9.7	11.0	8.9	13.5
PG-19	Accuracy	43.0	46.4	46.6	46.3	45.8	45.7	45.39	45.7	47.0
ScrollsQasper	ROUGE-L	2.9	5.0	4.9	4.5	4.1	4.5	5.4	5.1	7.2
ScrollsQuality	Accuracy	27.8	26.3	26.7	26.1	26.7	26.1	25.6	26.5	27.9
XLSum	ROUGE-L	8.5	13.6	11.8	12.4	12.3	11.8	14.2	14.4	15.9

Table 11: Results of long-context tasks evaluated by Flan 1B with different Transformer models and different memory segment configurations.

D.3. GPT-3 Tasks

We evaluated all model sizes on the tasks reported by GPT-3 (Brown et al., 2020). The results are shown in Table 12.

Dataset	Metric	BSWA 1B	FAM 1B	BSWA 8B	FAM 8B	BSWA 24B	FAM 24B
GPT-3 Rank	Mean	60.2	61.0	72.8	74.0	78.2	78.5
ANLI R1	Rank Acc.	42.2	42.5	66.1	64.8	70.8	78.3
ANLI R2	Rank Acc.	37.7	37.8	50.6	50.3	59.6	65.2
ANLI R3	Rank Acc.	39.4	38.6	50.5	51.3	60.1	62.8
ARC Challenge	Rank Acc.	37.6	37.9	57.0	56.6	62.8	63.9
ARC Easy	Rank Acc.	69.1	69.7	81.6	81.6	87.6	87.2
BoolQ	Rank Acc.	74.7	74.2	85.6	85.8	89.1	89.9
CB	Rank Acc.	73.2	78.6	67.9	91.1	100.0	82.1
COPA	Rank Acc.	71.0	78.0	90.0	91.0	91.0	93.0
HellaSwag	Rank Acc.	55.6	56.8	76.1	76.5	83.0	83.1
MultiRC	Rank Acc.	70.2	68.0	80.2	80.8	85.2	85.1
OpenbookQA	Rank Acc.	49.8	51.6	57.6	58.8	64.4	66.6
PIQA	Rank Acc.	74.1	74.5	80.4	81.0	84.2	84.0
RACE-H	Rank Acc.	39.3	38.8	49.7	50.5	55.1	56.3
RACE-M	Rank Acc.	55.0	53.6	65.8	65.9	70.8	70.5
ReCoRD	Rank Acc.	80.3	80.7	89.5	89.6	91.2	90.0
RTE	Rank Acc.	63.2	65.0	87.0	87.0	86.3	87.7
StoryCloze	Rank Acc.	73.8	75.2	82.7	82.6	87.5	85.8
WiC	Rank Acc.	53.4	51.9	58.6	62.2	54.5	53.3
Winograd	Rank Acc.	73.3	74.4	88.6	85.7	88.3	89.0
Winogrande	Rank Acc.	61.1	61.9	78.2	77.5	82.3	84.1
WSC273	Rank Acc.	70.9	72.3	85.3	84.2	87.4	89.5
GPT-3 Gen	Mean	33.9	34.7	54.3	54.9	62.6	63.4
LAMBADA	Decode Acc.	63.0	65.1	81.0	81.2	83.4	83.9
Natural Questions	EM score	10.8	10.7	25.6	25.3	37.6	37.3
SQuADv2.0	EM score	44.7	45.7	68.9	72.2	79.1	79.2
TriviaQA	EM score	24.3	24.6	58.3	58.2	69.7	76.1
WebQuestions	EM score	26.5	27.4	37.8	37.5	43.3	40.5

Table 12: Comparison of performance of GPT-3 tasks (Brown et al., 2020) between BSWA and FAM across 1B, 8B, and 24B models.

D.4. Complexity

BSWA and FAM have memory and computational complexity of $O(L \times C)$, where C is the chunk size. Table 13 and Table 14 show memory and inference time through inference jobs on TPUv4. The most tokens in the experiment are pre-filled and generation is limited to 256 tokens.

Memory (GB)	26k	34k	66k	130k	258k
BSWA	4.4	4.8	6.4	9.6	16.0
FAM	4.5	4.9	6.5	9.7	16.1

Table 13: Memory Usage Comparison: BSWA vs. FAM across varying sequence lengths.

E. Related Work

The Transformer architecture exhibits a quadratic complexity with respect to context length, a significant limitation. To address this, several research works have focused on approximating the attention mechanism. One approach involves sparse attention, where only a subset of important tokens are attended to, as seen in models like Sparse Transformer (Child et al., 2019), Big Bird (Zaheer et al., 2020), Reformer (Kitaev et al., 2020), Routing Transformer (Roy et al., 2021), and

Process secs	26k	34k	66k	130k	258k
BSWA	16.2	20.9	40.2	76.8	154.1
FAM	16.5	21.2	40.7	77.2	154.9

Table 14: Processing seconds Comparison: BSWA vs. FAM across varying sequence lengths.

TOVA (Oren et al., 2024). Linear approximation methods offer an alternative, seeking to circumvent the quadratic complexity by altering attention calculations, as exemplified by Linformer (Wang et al., 2020), Linear Transformer (Katharopoulos et al., 2020), Performer (Choromanski et al., 2020), and Nyströmformer (Xiong et al., 2021). Finally, some research explores entirely different sequence-to-sequence architectures as replacements for attention-based Transformers, including MLP-mixer (Tolstikhin et al., 2021), State Space Models (Rangapuram et al., 2018), S4 (Gu et al., 2021), Mamba (Gu & Dao, 2023), and RWKV (Peng et al., 2023).

Global Workspace Theory (GWT) (Baars, 2005) is a leading theory of consciousness. According to GWT, the human brain possesses a global workspace where various modules, such as sensory input, memories, and internal representations, converge. The attention mechanism in brain acts as a spotlight, focusing on specific inputs among the multitude, and transforming this unconscious activity into conscious awareness. These "momentarily active, subjectively experienced" events are then stored in working memory. TransformerFAM draws inspiration from GWT, adopting its principle of a unified attention mechanism for processing homogenous, heterogeneous, and feedback data.

F. Attention Visualization

Figure 12 depicts the attention map for each head in each layer of a 1B model. FAM is prepended and is located at the bottom left corner. The bright spots along the left edge represent the block inputs attending to FAM, while the bright spots along the bottom edge represent FAM compressing the corresponding block. Overall, the block inputs actively reference FAM, while FAM compresses only the selective inputs.

(a) 0th Layer, 2nd Head

(b) 1th Layer, 2nd Head

(c) 3rd Layer, 5th Head

(d) 4th Layer, 10th Head

(e) 9th Layer, 7th Head

(f) 10th Layer, 1st Head

(g) 13th Layer, 1st Head
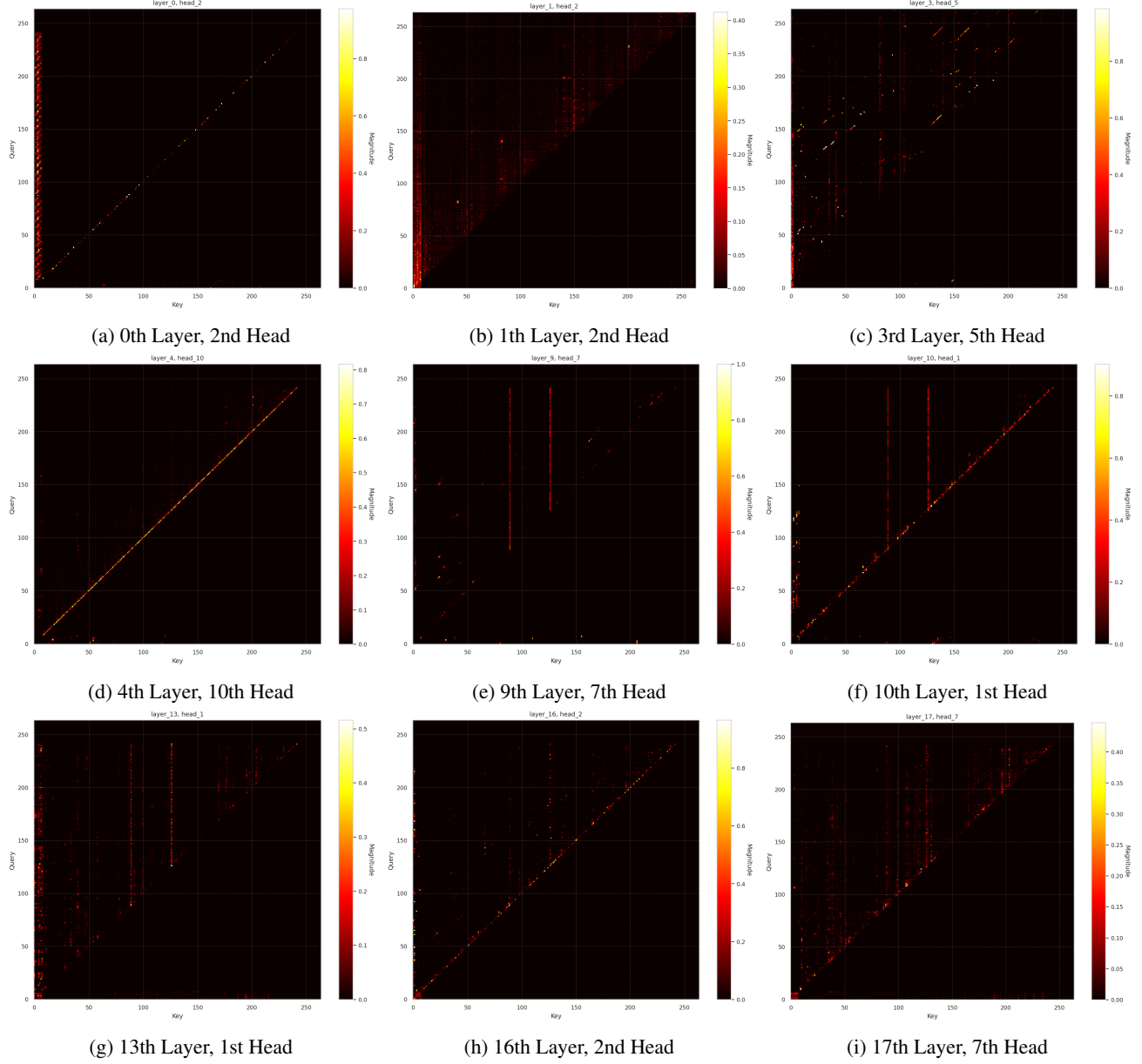
(h) 16th Layer, 2nd Head

(i) 17th Layer, 7th Head

Figure 12: Selected attention map of TransformerFAM with block size of 256 and FAM length of 8. The vertical axis of the attention map represents the Query, and the horizontal axis represents the Key. The FAM is prepended, and is located at the bottom left corner.