Transformers are Multi-State RNNs

Matanel Oren[*,H]   **Michael Hassid**[*,H,M]   **Nir Yarden**[H]

Yossi Adi[H,M]   **Roy Schwartz**[H]

[H]The Hebrew University of Jerusalem   [M]FAIR, AI at Meta

{matanel.oren,michael.hassid}@mail.huji.ac.il

Abstract

Transformers are considered conceptually different from the previous generation of state-of-the-art NLP models—recurrent neural networks (RNNs). In this work, we demonstrate that decoder-only transformers can in fact be conceptualized as unbounded multi-state RNNs—an RNN variant with unlimited hidden state size. We further show that transformers can be converted into *bounded* multi-state RNNs by fixing the size of their hidden state, effectively compressing their key-value cache. We introduce a novel, training-free compression policy—**T**oken **O**mission **V**ia **A**ttention (TOVA).[1] Our experiments with four long range tasks and several LLMs show that TOVA outperforms several baseline compression policies. Particularly, our results are nearly on par with the full model, using in some cases only ⅛ of the original cache size, which translates to 4.8X higher throughput. Our results shed light on the connection between transformers and RNNs, and help mitigate one of LLMs' most painful computational bottlenecks—the size of their key-value cache.[2]

1 Introduction

Not so long ago, transformers (Vaswani et al., 2017) replaced recurrent neural networks (RNNs; Elman, 1990) as the go-to architecture for NLP. Transformers are considered conceptually different than RNNs; they have direct access to each token representation in the sequence, while RNNs maintain a recurring state of previous inputs. Recently, *decoders* became a dominant transformer variant for large language models (LLMs; Brown et al., 2020; Touvron et al., 2023a; Jiang et al., 2023). These typically generate their output autoregressively—the generation of each token representation depends on
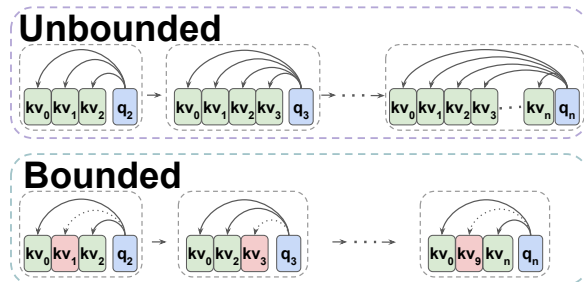

Figure 1: Top: transformers can be thought of as unbounded multi-state RNNs (MSRNNs), with the key-value vectors corresponding to a multi-state that dynamically grows infinitely (green elements). Bottom: transformers can be converted to *bounded* MSRNNs, which keep a fixed-size multi-state (here of size 2), by dropping one state (red elements) at each decoding step.

the key and value computation of previous tokens.[3]

In this work, we demonstrate that the autoregressivity of transformers aligns with the core principle of RNNs—preserving a state from one step to the other. We formally redefine decoder-only transformers as *multi-state* RNNs (MSRNN)—a generalized version of RNNs with multiple states, each corresponding to a history token. Importantly, as the number of tokens grows with each decoding step, transformers correspond to MSRNNs with an *unbounded* number of states (Fig. 1, top).

We then show that transformers can be compressed into *bounded* MSRNNs by limiting their number of states (Fig. 1, bottom). This process requires a compression policy for selecting the states to retain. While existing methods, e.g., windowed attention (Wang et al., 2019), can be cast as such policies, we propose a novel policy, TOVA, which retains the states with the highest attention scores.

We experiment with four long range tasks, sev-

---

[3]These previous computations are often cached for efficiency purposes, referred to as KV caching (Radford et al., 2019; Pope et al., 2023). We note that the arguments we make in this work apply similarly to non-cached implementations.

eral leading LLMs, and a few baseline compression policies. Our results show that TOVA outperforms all baselines in all setups. Further, using TOVA can match the performance of the full (uncompressed) model using as little as ⅛ of the full model multistate, which leads to a throughput increase of up to 4.8X. Finally, TOVA allows running on dramatically longer contexts, up to 70K tokens.

We finish by analyzing the states kept in memory by TOVA, and the tokens they correspond to. Unlike previous work (Xiao et al., 2024; Zhang et al., 2023), we observe that not all recent tokens are important to retain, and some may be safely dropped. We also show the importance of keeping the very first token in the sequence, as well as other, perhaps surprising tokens like possessive endings.

Our findings shed light on the connection between transformers and RNNs. They also help mitigate the LLM memory bottleneck during decoding, which directly translates to higher throughput.

2 Background

2.1 RNNs

Recurrent Neural Networks (RNNs; Elman, 1990) process sequential data recurrently. In the most general form, each layer l (often called a *cell*) is modeled as a function f_{RNN}^l that receives at time t two inputs: x_t^l, a representation of the current token, and h_{t-1}^l, the hidden state from the previous step. It then outputs two values: x_t^{l+1}, an updated token representation, and h_t^l, a new hidden state:

$$x_t^{l+1}, h_t^l = f_{\text{RNN}}^l(x_t^l, h_{t-1}^l) \qquad (1)$$

h_t^l is used for the recurrent computation over the next token x_{t+1}^l, while x_t^{l+1} is used as input to the next layer. It is common, though not necessary, to set $x_t^{l+1} := h_t^l$, i.e., the input for the following layer and the hidden state are the same.

2.2 Transformers

Transformers (Vaswani et al., 2017) process sequential data non-recurrently. A transformer layer f_{TRANS}^l takes as input a sequence of token representations of hidden size d: $X^l = (x_1^l, ..., x_t^l)^T \in \mathbb{R}^{t \times d}$ and returns a transformed representation:

$$X^{l+1} = f_{\text{TRANS}}^l(X^l) = \text{FF}^l\big(\text{SelfAttn}^l(X^l)\big) \quad (2)$$

Each transformer layer consists of two main components: self-attention (SelfAttn^l) and Feed-

Forward (FF^l).[4] The former operates over the entire sequence, while the latter on each token individually. Self-attention projects the input into three matrices: $Q^l, K^l, V^l \in \mathbb{R}^{t \times d}$, and computes:

$$X_{attn}^l = \text{Attn}(Q^l, K^l, V^l) \qquad (3)$$
$$= \underbrace{\text{Softmax}\big(Q^l \cdot (K^l)^T\big)}_{A^l} \cdot V \qquad (4)$$

where $A^l \in \mathbb{R}^{t \times t}$, the attention matrix, computes the interactions between tokens within a sequence.

In this work we focus on transformer **decoders**, which mask the upper triangular part of the attention matrix to perform next-token prediction. During decoding, it is common to cache the K, V matrices to avoid recomputing previous tokens.

3 Transformers as Multi-State RNNs

We start by formally defining a new RNN variant, Multi-State RNN (MSRNN; Sec. 3.1). We then show that transformers can be viewed as MSRNNs with an unbounded number of states (Sec. 3.2), and that their number of states can be bounded by applying a compression policy (Sec. 3.3). We finish by discussing LLMs as MSRNNs (Sec. 3.4).

3.1 Multi-State RNNs

We define an MSRNN as an RNN with a state *matrix* instead of a vector: $H_t^l \in \mathbb{R}^{g(t) \times d}$. The MSRNN equation corresponding to Eq. (1) is:

$$x_t^{l+1}, H_t^l = f_{\text{MSRNN}}^l(x_t^l, H_{t-1}^l) \qquad (5)$$

We can interpret each row of H_t^l as a single-state, allowing us to think of H_t^l as a multi-state matrix.[5]

The size of H_t^l is parameterized by a function g. Setting $g(t) = 1$ for all t reduces an MSRNN to a standard (single-state) RNN. Setting $g(t) \leq k$ for a constant k restricts it to a *bounded* memory capacity. If g is unbounded in t, the MSRNN state can have *unbounded* capacity.

3.2 Transformers are Unbounded MSRNNs

Consider the case where $g(t) = t$, i.e., the number of states equals the number of input tokens in the current time-step. In this setup, we can view

---

[4]Layer normalization, skip connections, and multiple attention heads are omitted for brevity.

[5]We could unroll the matrix and define it as a single vector in $\mathbb{R}^{g(t) \cdot d}$ and use the traditional RNN terminology, but we find it more convenient to think of it as a matrix.

a transformer as an unbounded MSRNN, where $H^l_t = (K^l_t, V^l_t)$ and the layer computation is:

$$(K^l_t, V^l_t) = \left(\begin{pmatrix} K^l_{t-1} \\ k^l_t \end{pmatrix}, \begin{pmatrix} V^l_{t-1} \\ v^l_t \end{pmatrix} \right) \qquad (6)$$

$$x^{l+1}_t = \mathrm{FF}^l\big(\mathrm{Attn}^l(q^l_t, K^l_t, V^l_t)\big) \qquad (7)$$

where q^l_t, k^l_t, v^l_t are the self-attention projections of x^l_t, and each state of (K^l_t, V^l_t) corresponds to a specific token. Combined, we get the MSRNN equation for transformers:

$$x^{l+1}_t, (K^l_t, V^l_t) = f^l_{\mathrm{TRANS}}\big(x^l_t, (K^l_{t-1}, V^l_{t-1})\big) \quad (8)$$

3.3 Converting Transformers into Bounded MSRNNs

Transformers can be converted into bounded MSRNNs by setting $g(t) = \min(t, k)$ for some k. When t exceeds k, a compression policy should be applied in order to fit the multi-state to into the bounded memory.

Interestingly, several existing KV cache compression methods, e.g., windowed attention (Wang et al., 2019) and H$_2$O (Zhang et al., 2023), can be seen as such compression policies, see Sec. 5.1.

3.4 LLMs as MSRNNs

LLMs are generally built as transformer decoders. As such, they are, on the one hand, unbounded MSRNNs (Sec. 3.2). On the other, they are trained with a fixed context length, and often struggle at extrapolating beyond it (Press et al., 2022), and thus may be considered bounded.

We argue that LLMs are indeed *unbounded*: At inference time, they can process any number of tokens, and are limited only by the available memory. In addition, both at training and inference time, they accumulate token representations into their multi-state without dropping any from their memory. Thus, as memory compression is the fundamental feature of bounded MSRNNs, LLMs should be conceptualized as unbounded. Interestingly, we later show that despite their unbounded capacity, they often act in practice as bounded MSRNNs.

4 TOVA: Token Omission Via Attention

Converting an unbounded MSRNN to a bounded one requires a state-compression policy (Sec. 3.3). We introduce TOVA—a novel, training-free policy for doing so (Fig. 2). After the multi-state reaches the capacity limit, TOVA drops at each decoding step the token with the lowest attention score. Formally, when $t > k$ and assuming j is the state with
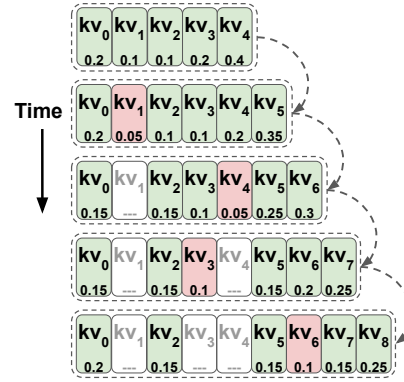

Figure 2: The TOVA policy keeps a fixed-size multi-state (green cells). At each decoding step (different rows), the state with the lowest attention score is omitted (red cells, which become transparent in subsequent steps).

the lowest attention score, TOVA applies the following over the multi-state (K^l_t, V^l_t) from Eq. (6):

$$(K^l_t, V^l_t) = \left(\begin{pmatrix} K^l_{0:j-1} \\ K^l_{j+1:k} \end{pmatrix}, \begin{pmatrix} V^l_{0:j-1} \\ V^l_{j+1:k} \end{pmatrix} \right) \qquad (9)$$

TOVA computes the attention scores of each head separately, and can thus retain different tokens at different heads. In practice, preliminary results show that averaging the attention scores across the heads of a given layer is superior to considering each head individually (App. A). See Alg. 1 for a torch-like implementation of TOVA.

5 Experimental Setup

We aim to check whether transformer LLMs converted into bounded MSRNNs can match the performance of the full model (an unbounded MSRNN; Sec. 3.4). Below we describe our baseline compression policies (Sec. 5.1), the datasets (Sec. 5.2), and the LLMs we experiment with (Sec. 5.3).

5.1 Baseline Compression Policies

Below we describe previously proposed compression policies. We note that, to the best of our knowledge, we are the first to make the connection between these policies and RNNs. As our focus is on the capacity of off-the-shelf models, we only consider baseline policies that operate on pretrained LLMs and require no additional training. Section 8 discusses approaches that do require training.

Window This policy (Wang et al., 2019) implements a First In First Out (FIFO) strategy. When the multi-state reaches its capacity, the oldest state (i.e., the earliest token state) is discarded, such that only the most recent states are kept.

Alg. 1 A torch-like implementation of TOVA. Batch size=1 is assumed for simplicity.

```python
def TOVA(attn_weights, k_cache, v_cache, cache_max_size):
    # k_cache.shape and v_cache.shape are [attn_heads, num_kv, hidden_dim]
    attn_heads, num_q, num_kv = attn_weights.shape
    if num_kv <= cache_max_size:
        return
    # Average last query attention weights across heads:
    mean_attn_weights = mean(attn_weights[:,-1,:], dim=0)
    minimal_idx = argmin(mean_attn_weights) # get the index to drop
    k_cache = concat([k_cache[:, :minimal_idx], k_cache[:, minimal_idx+1:]], dim=1)
    v_cache = concat([v_cache[:, :minimal_idx], v_cache[:, minimal_idx+1:]], dim=1)
```

Window+i This policy uses a fixed window, but also retains the first i states, for some constant i. Previous work (Xiao et al., 2024; Han et al., 2024) has shown that Window+i strongly outperforms Window using as few as 1–4 early states.

H$_2$O Much like Window+i, this policy (Zhang et al., 2023) keeps a fixed window of recent tokens, as well as additional earlier tokens. Unlike Window+i, it dynamically selects the non-window tokens by aggregating the attention scores throughout the sequence, and keeping the ones with the highest aggregated scores. The number of non-window tokens is typically set as half of the multi-state size. Like TOVA, H$_2$O can operate head-wise or layer-wise. Preliminary results (App. A) indicate that both variants perform similarly, so we follow Zhang et al. (2023) and use the head-wise version.

Full model (topline) We use the full (un-bounded) model as our topline. Pretrained transformers struggle with sequences longer than their pretrained sequence length (Press et al., 2022). In order to make the most fair comparison, we feed the model with the full training sequence length of the particular LLMs we use, and use smaller multi-state sizes for the different compression policies.[6]

We note that the all baseline policies presented above introduce strong inductive biases; e.g., devoting a substantial part of the state towards the most recent tokens, and preferring tokens appearing early in the sequence.[7] In contrast, TOVA makes fewer assumptions: it neither fixes a window of recent token-states, nor favors early ones.

5.2 Long Range Evaluation

To trigger the different policies, we focus on long range evaluation. We employ three types of long-range evaluation: language modeling, long-range understanding, and text generation. See App. B for the prompts used for the different tasks.

Language modeling We report perplexity on the PG-19 test set (Rae et al., 2020), a widely used benchmark for evaluating long range language models (So et al., 2024; Hutchins et al., 2022; Chen et al., 2023). PG-19 is composed of 100 full-length books of average length of 70k tokens.

Long range understanding We consider two tasks from ZeroSCROLLS (Shaham et al., 2023), each focusing on a different aspect of long range understanding: (a) SQuALITY (Wang et al., 2022), a question focused summarization dataset; and (b) QASPER (Dasigi et al., 2021), a QA dataset based on the S2ORC dataset (Lo et al., 2020). QASPER can be considered a retrieval task, as answering its questions requires retrieving specific details from long texts. For SQaULITY, we report the geometric mean of ROUGE-1/2/L scores (based on the gold summary, see Shaham et al., 2023). For QASPER, we follow Dasigi et al. (2021) and report F1 score.

Text generation We prompt the models to generate a long story. We sample 100 unique stories from each version of the model, using different seeds. As comparing between stories is hard, we employ GPT-4 as an evaluator (Chiang et al., 2023; Zhou et al., 2023). For each seed, we compare the two generated stories by asking GPT-4 which is better, reporting the average win rate for each approach. For further implementation details, see App. C.

5.3 Models

For language modeling, we experiment with three leading transformer decoder LLMs families, each offering a ~7B parameter version: LLaMA-2 (Touvron et al., 2023b), Mistral (Jiang et al., 2023) and Yi (Young et al., 2024). For long range understanding tasks, we consider three fine-tuned LLMs,

---

[6]In Sec. 7.2 we also report extrapolation experiments.

[7]Note that H$_2$O aggregates the attention weights, which favors initial tokens, as they accumulate more attention scores as the sequence progresses.
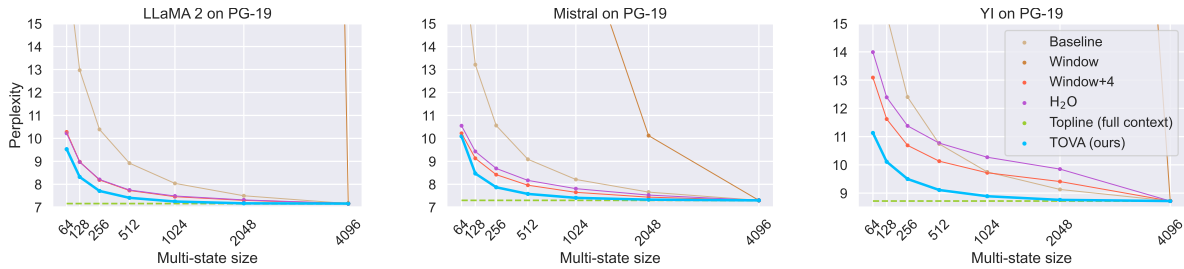
Figure 3: Perplexity results for the PG-19 test set. TOVA outperforms all other policies in all multi-state sizes, while maintaining comparable results to the full context topline using ⅛ of the context size.
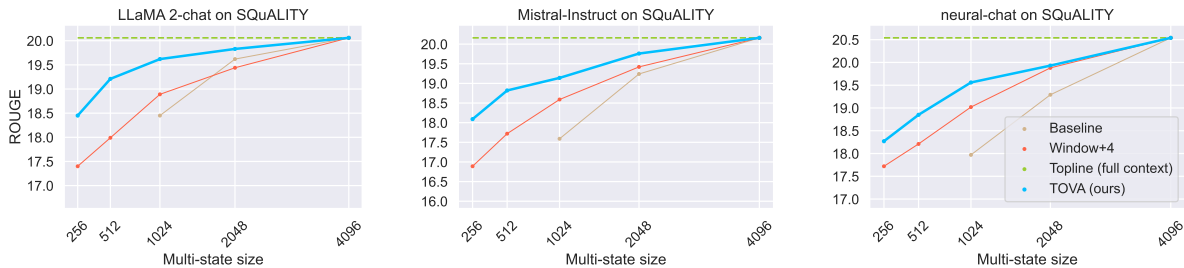

Figure 4: Geometric mean of ROUGE-1/2/L for SQuALITY. TOVA achieves within one point of the topline using $1/8 - 1/4$ of the multi-state size, while outperforming all other policies.
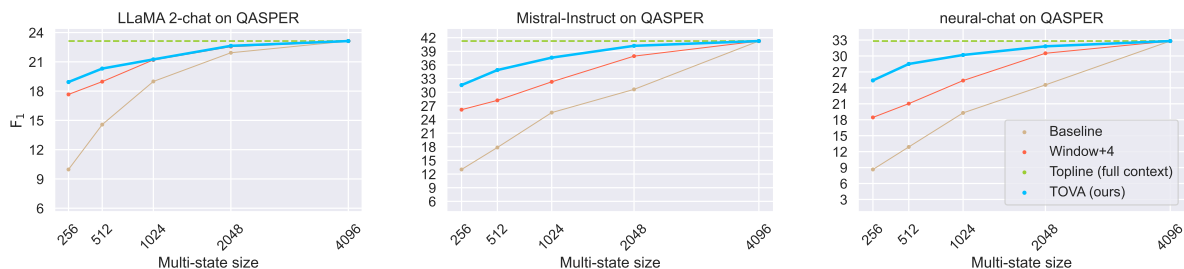

Figure 5: F1 score over QASPER benchmark. TOVA outperforms both baselines, but requires a half of the full multi-state size for obtaining comparable results to the topline.

which have been shown to excel in instruction tasks: LLaMA-2-chat (Touvron et al., 2023b), Mistral-Instruct (Jiang et al., 2023) and neural-chat (Lv et al., 2023). For text generation, we use Mytho-Logic (Padar, 2023), a LLaMA-2-13B version fine-tuned for story generation.

For all models and tasks, we use the full training sequence length of 4,096 tokens. For language modeling, we split the texts into chunks of length 4,096, and apply efficient masking (see App. D). For the language understanding tasks, we truncate the end of the example (excluding prompt) if it exceeds 4,096 tokens, as in Shaham et al. (2023).

6 Results: Pretrained Transformers Often Act as Bounded MSRNNs

6.1 Language Modeling

We evaluate our base models over the language modeling task using the following policies: Win-

dow, Window+4, H_2O and our TOVA policy.[8] As an additional baseline, we run the models with a smaller sequence length, while not applying compression, which corresponds to an unbounded MSRNN with a shorter sequence length. We examine multi-state sizes in exponential scales of 2^j for $j \in \{6, 7, \ldots, 12\}$ (2^{12}=4,096).

Figure 3 shows the perplexity results on PG-19. In all cases, TOVA performs within 0.4 points of the topline using one eighth of the full context length. Our results are consistently better than all baselines, which require at least half of the full context length to reach the full model results. Based on our results, we consider two policies for the other tasks: TOVA and Window+4, our best baseline.

6.2 Long Range Understanding

We evaluate instruction-tuned LLMs on SQuAL-ITY and QASPER.[9] As an additional baseline, we

---

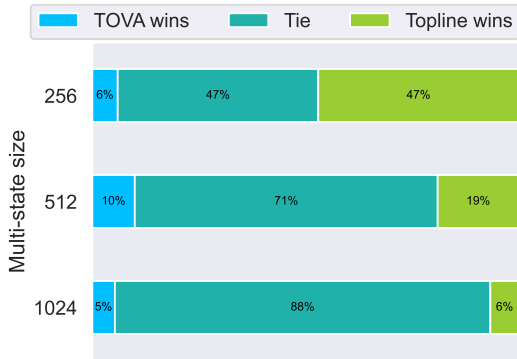[8]We ablate other policies in App. A.

[9]Base LLMs numbers are reported in App. E.

Figure 6: GPT-4 preference over stories generated by the full model and using TOVA.

Multi-state size	256	512	1,024	2,048	4,096 (full)
Memory (Gig.)	0.15	0.28	0.56	1.11	2.18
Maximal batch	139	70	35	17	8
Rel. throughput	8.5	4.8	3.1	1.7	1

Table 1: TOVA substantially reduces memory requirements (first row), and accordingly allows for increased batch size (second) and throughput (third row). The first row is with a batch size of 1; the second row shows the maximal batch size for decoding the same number of tokens on a single V100 machine. The last row is the overall decoding throughput when the maximum batch size is employed, relative to a full multi-state size.

present the model with a truncated version of the example according to the MSRNN capacity. E.g., for a multi-state of size k, the example is truncated to k tokens (including the prompt). As multi-state sizes, we consider 2^j for $j \in \{8, 9, \ldots, 12\}$.

Results for SQuALITY are shown in Fig. 4. TOVA consistently outperforms all baselines across all setups. As in language modeling, TOVA requires a quarter (Mistral and Yi) or even one eighth (LLaMA-2) of the full context to reach within one point of the topline.

Figure 5 shows the QASPER results. The gap between TOVA and the baselines is large, in some cases reaching beyond 5 F1 points. Nonetheless, here TOVA needs half of the full context to perform within one F1 point of the topline.

6.3 Text Generation

We compare TOVA to the topline on text generation. We first note that limiting the multi-state size makes the generated text shorter: the average story length for the full model is 1,566 tokens. This value is kept for a multi-state size of 1,024, but drops to 1,503 with 512 tokens and to 1,361 with 256 tokens.

Figure 6 shows the evaluation results of the stories using GPT-4. Using 256 tokens our policy losses to the topline in 47% of cases, while winning or tying in the remaining cases. This loss rate decreases substantially to 19% with 512 tokens and further to only 6% with 1,024 tokens. Importantly, our policy is also preferred over the topline in 5–10% of the cases in all multi-state sizes considered.

6.4 Discussion

Our results indicate that transformer decoder LLMs often behave empirically as bounded MSRNN: in 2/4 tasks, using TOVA with as little as ⅛–¼ of the multi-state size yields comparable results to the

topline. The other two tasks, text generation and retrieval QA, seem to require larger multi-state sizes, though still maintain comparable performance using half of the full multi-state. This suggests that the conversion of a transformer into an RNN reintroduces the inherent challenges associated with RNNs, as they encounter difficulties with retrieving distant information (Hochreiter and Schmidhuber, 1997; Arjovsky et al., 2016; Jelassi et al., 2024).

7 Analysis

We analyze TOVA in terms of memory and throughput efficiency (Sec. 7.1), extrapolation (Sec. 7.2), and the tokens frequently kept by it (Sec. 7.3). Throughout the section we use LLaMA-2-7B.

7.1 TOVA is Time- and Memory-Efficient

As discussed in Sec. 2.2, caching the K, V matrices in transformer autoregressive decoding is common in current frameworks. When employing TOVA, or any similar cache compression policy, these matrices are compressed, which leads to a proportional reduction in memory requirements (Tab. 1, first row). Importantly, beyond the the KV cache, the LLM decoding memory consumption is determined by two additional factors: the model size (e.g., number of layers, hidden size), and the batch size. As the former is fixed, caching effectively limits the inference batch-size. Table 1 presents the maximum batch size that can be used in our setup for decoding sequences of length 4,096, along with the corresponding throughput (tokens/sec) while decoding 512 sequences (totaling 2M tokens). TOVA with a multi-state of 512, which performs comparably to the full (4,096) model (Sec. 6), allows almost a 9X increase in batch size, and a corresponding
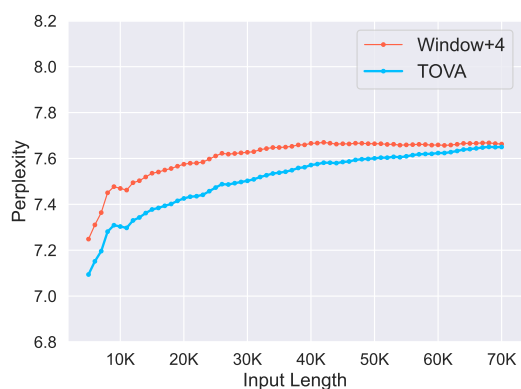
Figure 7: TOVA successfully extrapolates well beyond pretraining context length, and outperforms Window+4. Each point is the average over all previous tokens.
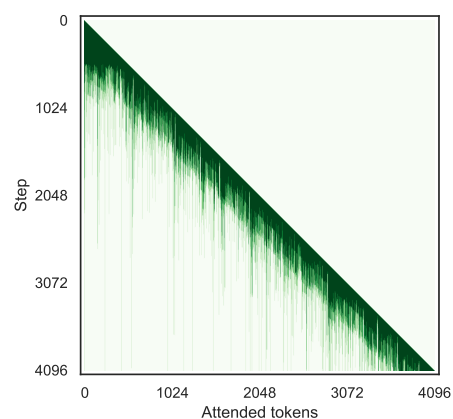

Figure 8: The tokens kept by TOVA in the final layer of LLaMA-2-7B on one PG-19 example. Rows represent decoding steps, while columns represent the attended tokens. The diagonal pattern corresponds to recent tokens, and the vertical lines match older tokens that are retained across steps.

speedup of 4.8X compared to the full model.

7.2 Extrapolation with TOVA

We further test the ability of bounded MSRNNs in handling longer texts, i.e., beyond the training sequence length. Using TOVA, this requires adapting the positional encoding of cached tokens to avoid values unseen during training. To do so, we compress the gap g between adjacent token representations to be $ln(ln(g))$,[10] while keeping g fixed if $g \leq 10$ to retain local sensitivity. E.g., for adjacent tokens with positions $(i, i + g)$, the new positions will be $(i, i + ln(ln(g)))$, or $(i, i + g)$ if $g \leq 10$.

We report the average perplexity on the first 70K tokens of all PG-19 books with at least that number of tokens (52 books in total). We use a multi-state size of 512. As models struggle to extrapolate to such long contexts, we only compare TOVA with Window+4, which has been shown to support such contexts (Xiao et al., 2024; Han et al., 2024). Our results (Fig. 7) show that TOVA extrapolates well up to 70K tokens with a similar performance to the shorter contexts (less than 0.5 PPL points difference), while outperforming Window+4.

7.3 Which Tokens Matter?

Our results indicate that most token representations may be dropped from memory as generation progresses. We characterize the tokens frequently dropped by running TOVA on 31 PG-19 instances.

Recency is *not* all you need Much like most compression policies (Sec. 5.1), TOVA preserves recent tokens. Figure 8 illustrates the tokens kept

by TOVA in the final layer for one PG-19 example, using a multi-state size of 512.[11] We see a clear window trend, indicating the importance of recent tokens. Nonetheless, we also observe that many older tokens are kept. To quantify this, we compute the proportion of recent tokens of all tokens kept in the multi-state, averaged across examples, layers, and positions. We find that only 73–76% of the tokens are recent. This suggests that while recent tokens are important, they are far from sufficient. Importantly, unlike existing policies that handcraft the recent window (Xiao et al., 2024; Zhang et al., 2023), TOVA identifies it automatically. We turn to study which early tokens tend to be kept, considering two dimensions: position and content.

The first token matters Figure 9 shows the number of decoding steps each of the first 25 tokens is kept (averaged across layers and examples). As previously observed (Han et al., 2024; Xiao et al., 2024), we find that the first token is kept until the end of the sequence across all multi-state sizes. However, other early tokens are dropped far faster.

Not all tokens are equally kept As indicated by Fig. 8, some tokens last much longer than others. To further study it, we map each token to its part-of-speech tag using NLTK (Bird et al., 2009), and plot the tags that last longest in Tab. 2. Our results show that, as observed by previous work (Clark et al., 2019; Zhang et al., 2023; Ge et al., 2024), punctuation and other special symbols tend to be

---

[10]Preliminary experiments with other compression functions, e.g., $ln(g)$ and $sqrt(g)$ showed inferior results.

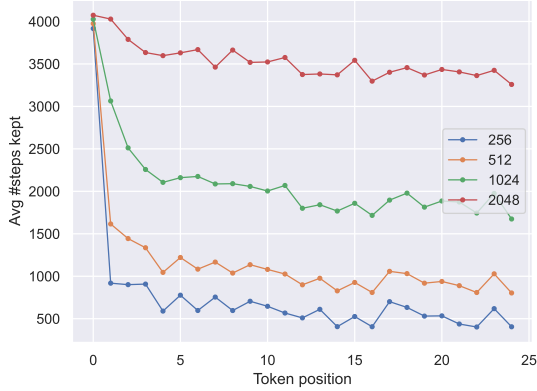[11]See App. F for the illustrations of all layers.

Figure 9: The average number of steps a token is kept in the multi-state when applying TOVA as a function of token position. Different lines are different multi-state sizes. The very first token is kept through the entire context, while next tokens are dropped far earlier.

Multi-state size Tag	256	512	1024	2048
Avg.	249	481	897	1537
POS	1134	1393	1736	2061
"	845	1101	1413	1774
$	329	724	1276	2123
)	379	670	1161	1558
.	350	645	1117	1677
NNPS	321	578	1042	1671
\n	303	550	969	1538

Table 2: Mean number of steps tokens are kept in the multi-state with TOVA, grouped by part-of-speech tags. Columns represent the multi-state size. Here we report the tokens kept the longest, see full table in App. G.

kept. However, we also identify other tokens that tend to stay longer, e.g., possessive endings (POS) and proper nouns (NNPS). Studying the role of these tokens is an important direction for future work.

8 Related Work

Transformers and RNNs Several works have tried to bridge the gap between RNNs and transformers. Hutchins et al. (2022) employed a hybrid approach that attends both to recent tokens and to further hidden states. Sun et al. (2023) substituted the self-attention layer with a convolution layer that can be applied recurrently. Peng et al. (2023) adjusted the self-attention layer to perform recurrence at inference. So et al. (2024) proposed a model with repeated layers to perform recurrent computations over the signal, rather than over time.

Most related to this work are Katharopoulos et al. (2020) and Peng et al. (2022). The former suggested that transformers can be used in a recurrent manner, and proposed a linear transformer for doing so. The latter presented transformers with bounded memory, showing that several transformer variants such as Linformer (Wang et al., 2020) and window attention can be interpreted as instances of their framework. Unlike us, these works treat the memory as a single state, without an explicit mapping from tokens to states. Moreover, unlike our approach, the works above require a dedicated training, and cannot operate on existing LLMs.

Limited KV cache Window attention (Wang et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020) and its variants (e.g., H_2O, Zhang et al., 2023; SCISSORHANDS, Liu et al., 2024) are simple ways of limiting the cache size in transformers. A recent followup work (Ge et al., 2024) showed that manually caching specific tokens like "." and "," further boosts H_2O performance. We showed that TOVA does so without manually selecting tokens (Sec. 7.3). Anagnostidis et al. (2023) introduced a learned approach over LLMs that limits the cache consumption of transformers. Yun et al. (2023) and Berchansky et al. (2023) proposed token pruning and token combining.

Concurrent to our work, Ren and Zhu (2024) suggested robustness measures to choose which states to drop; Brandon et al. (2024) showed that KV cache can be shared across layers; Yang et al. (2024) proposed a pyramid structure across layers to reduce cache size; Li et al. (2024) and Zandieh et al. (2024) suggested clustering the KV cache, and Kang et al. (2024) proposed to quantize and approximate it. None of these works drew a connection between RNNs and transformers.

New RNN variants Recent work aimed to revive RNNs in NLP. S4 (Gu et al., 2022) and its successors (Gupta et al., 2022; Mehta et al., 2023; Gu and Dao, 2023) elevate state spaces to form linear RNNs. Other work introduced RNN variants that train efficiently (Merity, 2019; Orvieto et al., 2023; Yang et al., 2023; Beck et al., 2024).

Simplifying transformers Previous work has shown that many transformer attention heads can be pruned (Michel et al., 2019; Li et al., 2021) or replaced with static weights (Hassid et al., 2022). Several works replaced the attention mechanism in transformers with efficient variants (Peng et al., 2021; Choromanski et al., 2021; Liu et al., 2021;

Lee-Thorp et al., 2022). We show that transformer decoders can be reduced to bounded MSRNNs.

9 Conclusion

In this work, we redefined decoder transformers as a form of multi-state RNNs (MSRNN) with an unbounded multi-state size. We then showed that they can be compressed to bounded MSRNNs by limiting the number of tokens they can handle at each decoding step.

We introduced TOVA, a conceptually simple compression method that selects which tokens to keep using their attention scores. We showed that TOVA is superior compared to existing compression policies; in many cases, TOVA performs comparably to the full (unbounded) model, while requiring ⅛–¼ of the multi-state size. TOVA also allows processing long inputs, up to 70K tokens.

Our findings shed light on the inter-working of transformers, and their connections to RNNs. They also have practical value—they can reduce the LLM cache size by up to 88% and increase throughput by 4.8X.

Limitations

Evaluating models on long text generation is computationally expensive and might limit others from reproducing our results. Further, the evaluation of such task is extremely complicated, even for humans. We therefore resort to GPT-4 to compare the output of our TOVA policy compared to the topline model (Sec. 6.3). We recognize that this is far from perfect, and will most likely not catch the full breadth of evaluating text quality. Finally, our evaluation framework focuses on English tasks. It is not unlikely that languages with more flexible word order will make different use of the attention mechanism, and thus might require a larger multi-state size.

Ethics Statement

Our work has the potential to dramatically reduce the memory footprint of transformer LLMs, thereby potentially increasing their adoption by users with limited hardware access.

This work does not collect any new data, and only uses open source models, and public data collected by other sources.

References

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2023. Dynamic context pruning for efficient and interpretable autoregressive transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xLSTM: Extended long short-term memory. arXiv:2405.04517.

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

Moshe Berchansky, Peter Izsak, Avi Caciularu, Ido Dagan, and Moshe Wasserblat. 2023. Optimizing retrieval-augmented reader models via token elimination. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1524, Singapore. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. 2024. Reducing transformer key-value cache size with cross-layer attention. arXiv:2405.12981.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. arXiv:2306.15595.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion

Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* Chat-GPT quality.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *ICLR 2024*.

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

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. LM-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, Mexico City, Mexico. Association for Computational Linguistics.

Michael Hassid, Hao Peng, Daniel Rotem, Jungo Kasai, Ivan Montero, Noah A. Smith, and Roy Schwartz. 2022. How much does attention actually attend? questioning the importance of attention in pretrained transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1403–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. In *Advances in Neural Information Processing Systems*.

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM. arXiv:2403.05527.

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

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2022. FNet: Mixing tokens with Fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4296–4313, Seattle, United States. Association for Computational Linguistics.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM knows what you are looking for before generation. arXiv:2404.14469.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to MLPs. In *Advances in Neural Information Processing Systems*, volume 34, pages 9204–9215. Curran Associates, Inc.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2024. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In *Advances in Neural Information Processing Systems*, volume 36.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Kaokao Lv, Wenxin Zhang, and Haihao Shen. 2023. Supervised fine-tuning and direct preference optimization on Intel Gaudi2.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2023. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations*.

Stephen Merity. 2019. Single headed attention RNN: Stop thinking with your head. arXiv:1911.11423.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Gryphe Padar. 2023. Mythologic-l2-13b.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2022. ABC: Attention with bounded-memory control. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7469–7483, Dublin, Ireland. Association for Computational Linguistics.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *Proc. of ICLR*.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. In *Proc. of MLSys*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Siyu Ren and Kenny Q. Zhu. 2024. On the efficacy of eviction policy for key-value constrained generative language model inference. arXiv:2402.06262.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989, Singapore. Association for Computational Linguistics.

David R. So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. 2024. Primer: searching for efficient transformers for language modeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NeurIPS '21, Red Hook, NY, USA. Curran Associates Inc.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller,

Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. LLaMA 2: Open foundation and fine-tuned chat models. arXiv:2307.09288.

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

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R. Bowman. 2022. SQuALITY: Building a long-document summarization dataset the hard way. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1139–1156, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. arXiv:2006.04768.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5878–5882, Hong Kong, China. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *Proc. of ICLR*.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024. PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3258–3270, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

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

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2024. Yi: Open foundation models by 01.AI.

Jungmin Yun, Mihyeon Kim, and Youngbin Kim. 2023. Focus on the core: Efficient attention via pruned token compression for document classification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13617–13628, Singapore. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Amir Zandieh, Insu Han, Vahab Mirrokni, and Amin Karbasi. 2024. SubGen: Token generation in sublinear time and memory. arXiv:2402.06082.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021. Curran Associates, Inc.

A Policy Ablation

We ablate all policies presented in Sec. 5.1 and several TOVA variants with the language modeling task. Specifically we examine: Window, Window+i for $i \in \{1, 4\}$, H_2O for both per layer and per head approaches and our TOVA policy for both per layer and per head approaches. We also combine TOVA with additionally fixing the first i tokens using $i \in \{1, 4\}$. We consider the same baseline policy as in Sec. 6.1. We use the LLaMA-2-7B as the backbone model.

Our results are presented in Tab. 3. As shown in Sec. 6.1 the Window policy fails, while the Window+1 and Window+4 policies maintain much better results (with Window+4 performing slightly better). The two H_2O policies (head/layer) produce similar results. Regarding our TOVA policies, the head version performs worse than former policies in most multi-state sizes, while the layer version outperforms all other policies. We attribute this difference to the more robust selection mechanism employed by the layer version, which requires agreement among all heads to determine the importance of specific tokens. Lastly, when we enhance our TOVA policy with the explicit preservation of i initial tokens, the results remain relatively unchanged, implying that our policy inherently retains the crucial tokens.

B Prompts

The prompts used for the different evaluations through this work are presented in Tab. 4.

C Details of Generation Evaluation

To evaluate the generated texts, using GPT-4, we use the gpt-4-0613 version. We drop cases where the model stops generating before reaching the memory limit, as both stories are identical. To account for GPT-4's positional bias (Wang et al., 2024), we present each pair of stories twice, alternating their positions, and only consider a win if the same approach is preferred in both cases.

D Experimental Details

All experiments are done using bfloat16 floating-point precision over Nvidia V100 GPUs. To effectively parallelize the language modeling task for all tokens in the sequence, we modify the attention mask to incorporate the different MSRNN policies presented in Sec. 3. Specifically, for Window and Window+i policies, we apply a static masking, as the reduced tokens are independent with respect to the attention computation. For H_2O and TOVA, we adjust the mask according to the attention weights of the relevant layer.

E Long Range Understanding with Base Models

Figures 10 and 11 show the results for base LLMs over the SQuALITY and QASPER benchmarks, respectively.

F Illustration of the Tokens Retained by TOVA

Figures 12 and 13 show illustrations of the tokens retained (X axis) at each step (Y axis) for every layer of LLaMA-2-7B, when applying TOVA over one PG-19 example. We use a multi-state size of 512.

G Full Part-of-Speech Tag Analysis

The full version of Tab. 2 is presented in Tab. 5.

Multi-state size / Policy	64	128	256	512	1024	2048	4096
Baseline	17.65	12.97	10.39	8.92	8.04	7.50	**7.16**
Window	4812.27	4025.01	3275.58	2184.62	1001.29	240.17	**7.16**
Window+1	10.20	8.97	8.22	7.76	7.50	7.33	**7.16**
Window+4	10.28	8.98	8.19	7.73	7.46	7.30	**7.16**
H_2O-head	10.22	8.97	8.21	7.75	7.49	7.32	**7.16**
H_2O-layer	10.20	8.97	8.22	7.76	7.50	7.33	**7.16**
TOVA-head	11.13	9.55	8.69	7.90	7.52	7.27	**7.16**
TOVA-layer	**9.53**	**8.32**	**7.71**	**7.41**	**7.25**	**7.17**	**7.16**
TOVA-layer+1	**9.53**	**8.31**	**7.71**	**7.41**	**7.25**	**7.17**	**7.16**
TOVA-layer+4	9.63	**8.33**	**7.72**	**7.41**	**7.25**	**7.17**	**7.16**

Table 3: Perplexity over the PG-19 set using varying multi-state sizes (maximal number of states used), while ablating several dimensions such as the number of recent tokens in Window+i policies and head vs. layer selection in H_2O and TOVA. Our TOVA policy dominates the table in all multi-state sizes.
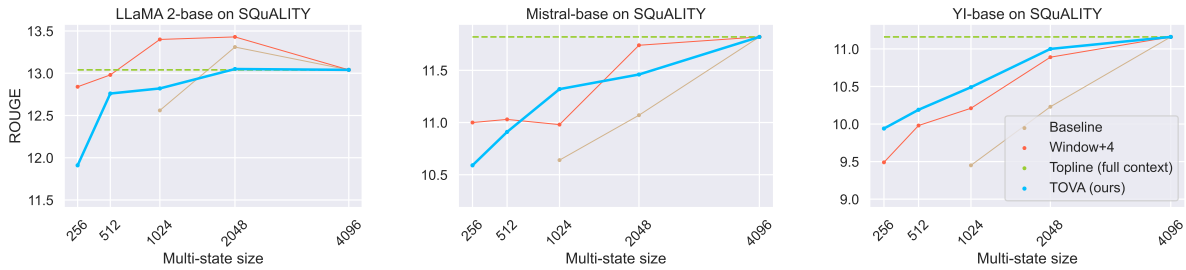

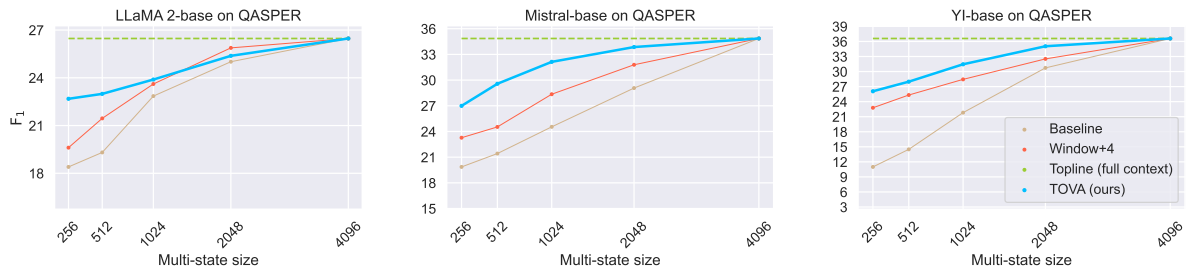Figure 10: Geometric mean of ROUGE-1/2/L for SQuALITY using the base LLMs.

Figure 11: F1 scores over the QASPER benchmark using base LLMs.

Task	Prompt
SQuALITY	{Story}
	Answer the question in a paragraph.
	Question:
	{Question}
	Answer:
QASPER	{Article}
	Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable".
	Question:
	{Question}
	Answer:
Story Generation	### Instruction: Write a very long story (at least 4,000 words). The story should include at least 20 named characters, spans 3 countries and 9 cities, at least 10 chapters and should have a lot of plot twists.
	### Response:
GPT- Evaluation	Help me decide which response is better given the prompt: {Prompt body for story generation} Which of the following responses is better (the responses are separated by '————————'):
	Response (A): {First Response}
	————————
	Response (B): {Second Response}
	Comparing these two responses, which response is better (A), (B) or (C) for equal quality? please select one and only one option, be as concisely as you can, using a single phrase.

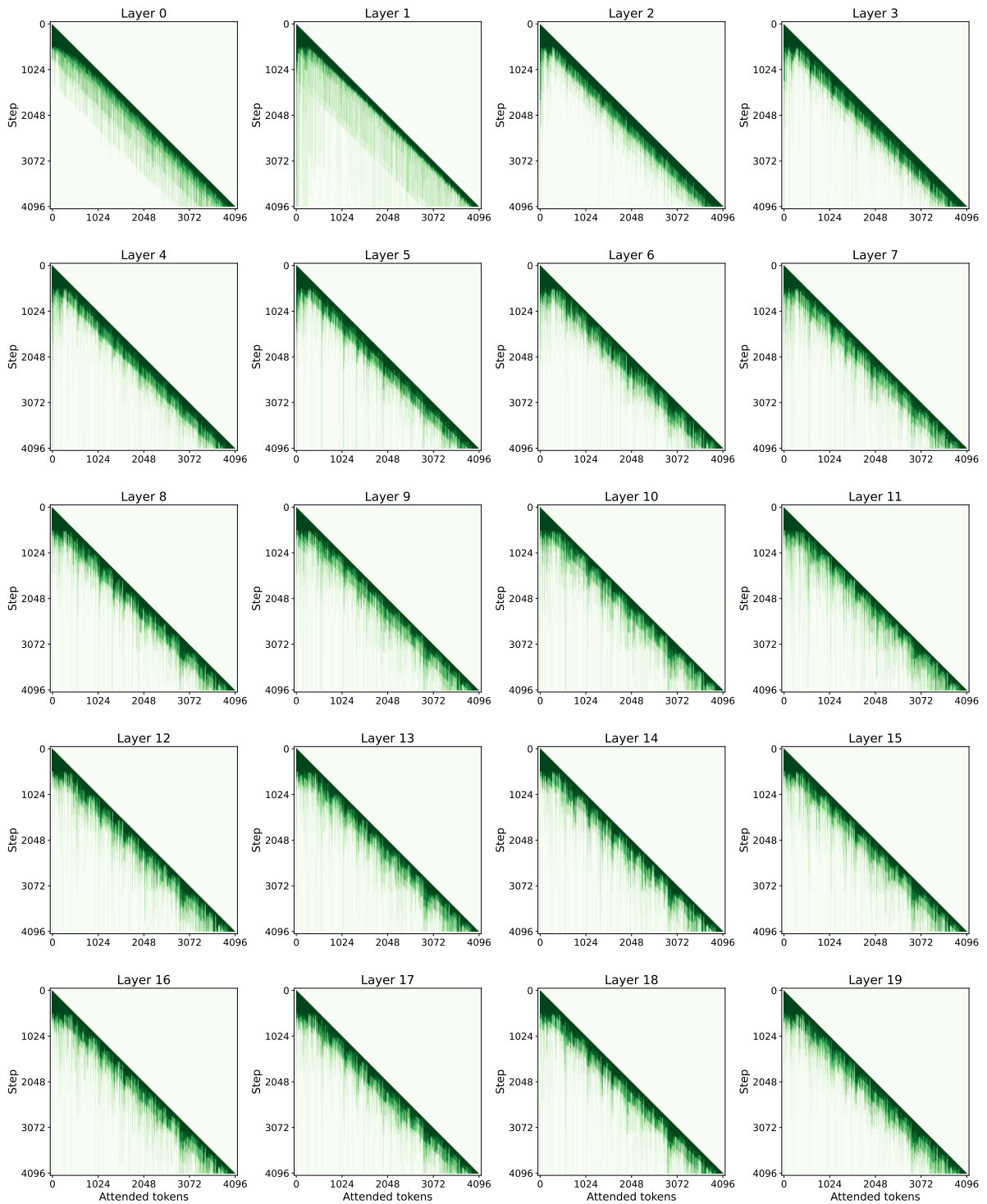Table 4: Prompts used for our experiments.

Figure 12: The full illustration corresponding to Fig. 8 of the tokens kept by TOVA for all layers of LLaMA-2-7B on one PG-19 example. Each row represents a decoding step, and each column is a token attended to. Layers 0–19.
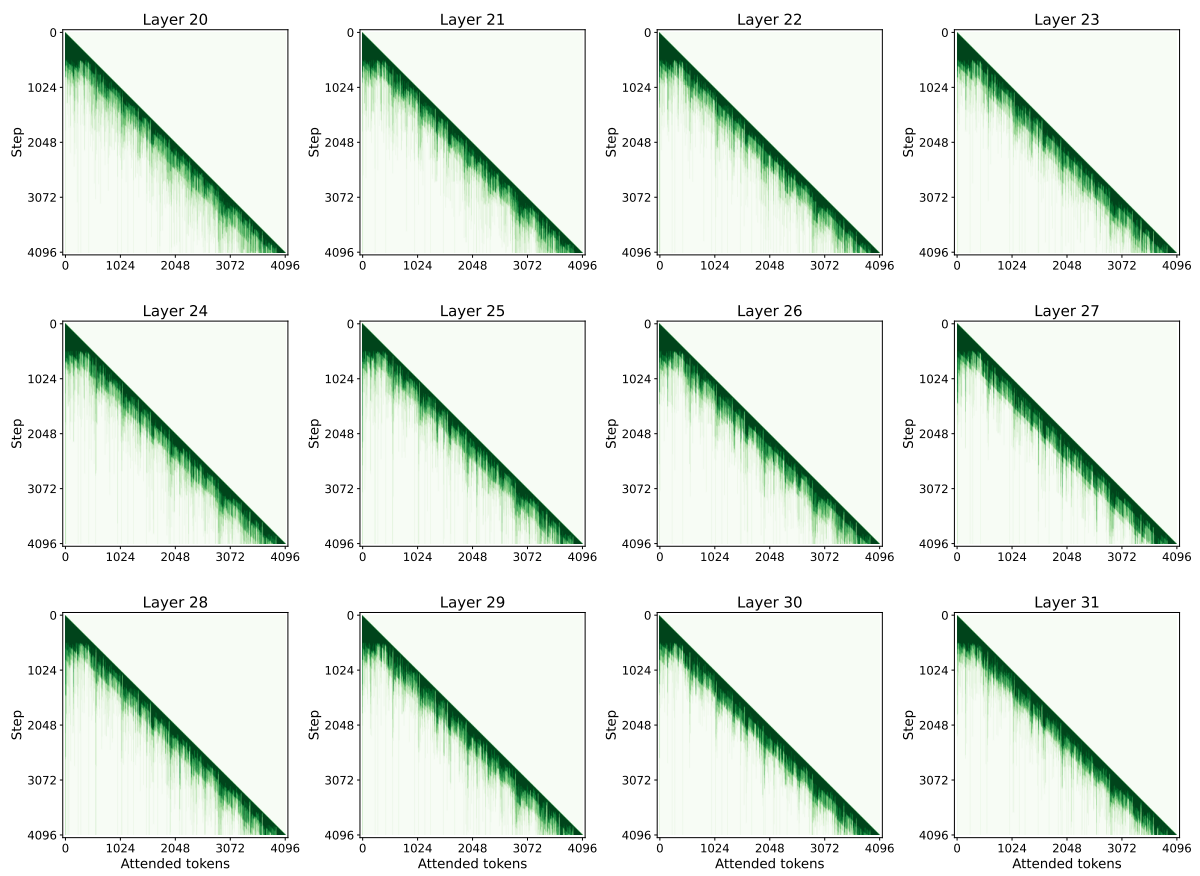
Figure 13: Continuation of Fig. 12 for layers 20–31.

Multi-state size	256	512	1024	2048
Tag				
Avg.	249	481	897	1537
POS	1134	1393	1736	2061
”	845	1101	1413	1774
$	329	724	1276	2123
)	379	670	1161	1558
.	350	645	1117	1677
NNPS	321	578	1042	1671
\n	303	550	969	1538
WP$	255	539	1121	1920
CD	301	537	940	1557
NN	270	527	983	1628
NNS	270	526	978	1618
NNP	270	517	951	1613
FW	253	511	903	1444
:	243	492	940	1570
JJ	240	480	918	1598
VBP	244	478	882	1504
JJS	220	475	953	1689
UH	233	474	870	1412
SYM	231	471	893	1482
WDT	223	462	903	1604
VBN	230	462	887	1549
EX	244	461	847	1461
RB	223	459	892	1566
,	236	453	840	1454
VBG	221	445	858	1523
RBS	210	441	878	1645
VBZ	219	440	844	1492
CC	217	437	862	1546
VBD	217	432	827	1493
VB	214	426	817	1457
PRP	217	424	794	1432
RP	207	417	811	1485
WRB	207	415	800	1502
WP	199	405	803	1506
JJR	195	403	782	1413
RBR	183	397	821	1566
PDT	181	391	756	1362
IN	190	385	760	1408
PRP$	189	383	745	1386
DT	190	379	734	1363
MD	177	378	754	1392
TO	182	368	734	1363

Table 5: Mean number of steps a token lasts, grouped by part-of-speech tags.