Cumulative Retention Autoregressive Models for Efficient Sequence Modeling

Abhinav Muraleedharan
University of Toronto
abhi98m@cs.toronto.edu

Abstract

While transformer-based autoregressive models have shown remarkable performance in sequence transduction problems across many domains such as text, audio, images, the complexity of training and inference of large-scale Transformer models is quadratic in sequence length and hence computationally expensive. Many novel architectures like state space models, and linear attention models have been proposed to address the computational inefficiencies of Transformer-based models but all of them still lag behind Transformers in terms of performance on key reasoning benchmark tasks (e.g., five-shot MMLU, Phonebook Lookup). In this work, we introduce Cumulative Retention Autoregressive Models[1] to address limitations of attention-based Transformer architectures. Unlike Transformers, the computational cost of training Cumulative Retention Autoregressive Models is linear with respect to sequence length $\mathcal{O}(N)$. Compared to State Space models, Cumulative Retention Autoregressive models involve nonlinear Transformations and hence they are suitable for complicated sequence modelling tasks.

1 Introduction

Autoregressive models, particularly those based on the Transformer architecture [12], have become the foundation for state-of-the-art performance across a wide range of sequence transduction tasks. From natural language processing (NLP) [8] to speech recognition [1] and even image generation [3], Transformers have demonstrated remarkable capability in capturing complex dependencies within sequential data. Their success is largely attributed to the self-attention mechanism [2, 7], which allows them to model long-range dependencies effectively. However, this expressiveness comes at a significant computational cost: both training and inference have a complexity of $\mathcal{O}(N^2)$ with respect to sequence length, where N is the number of tokens in the input sequence. This quadratic growth severely limits the scalability of Transformers, particularly in tasks involving long sequences.

Efforts to reduce the computational cost of Transformers have led to the development of several novel architectures, including state space models (SSMs) [4] and linear attention

[1]https://github.com/Abhinav-Muraleedharan/cram.git

models [6]. These architectures promise more efficient scaling, reducing the complexity of attention from quadratic to linear or near-linear in sequence length. State space models, for example, leverage the mathematical framework of linear dynamical systems to achieve $\mathcal{O}(N)$ time complexity. Similarly, linear attention mechanisms approximate the attention matrix, reducing its size and computational footprint. Despite these advancements, these models consistently underperform Transformers on key reasoning and comprehension benchmarks, such as the five-shot MMLU task [5] and the Phonebook Lookup task [9].

The gap in performance between these efficient models and Transformers is especially evident in tasks that require complex reasoning. Transformers' attention mechanism excels in handling nonlinear dependencies and intricate contextual interactions, which are critical for reasoning tasks. In contrast, many efficient models rely on linear operations, which can struggle to capture such rich interactions, particularly in multi-hop reasoning or memory-intensive tasks.

In this paper, we introduce Cumulative Retention Autoregressive Models (CRAMs), a novel architecture designed to bridge the gap between computational efficiency and reasoning capability. CRAMs maintain linear time complexity $\mathcal{O}(N)$ with respect to sequence length, akin to state space models, but crucially incorporate nonlinear transformations within the model's structure. This combination of efficiency and nonlinearity enables CRAMs to perform well on reasoning-heavy tasks while still scaling to long sequences.
Our key contributions can be summarized as follows:

- We propose Cumulative Retention Autoregressive Models (CRAMs), an architecture that combines the efficiency of linear-time models with the expressive power of nonlinear transformations.
- We show that CRAMs outperform state space and linear attention models on key reasoning benchmarks, including long-range arena tasks, narrowing the performance gap with Transformers.
- We provide a detailed analysis of the computational efficiency and scalability of CRAMs, demonstrating their ability to handle long sequences more efficiently than traditional Transformers without sacrificing performance.

2 Background and Related Work

The Transformer architecture [12] has become the standard backbone for autoregressive models across diverse sequence modeling tasks, thanks to its ability to capture long-range dependencies through the self-attention mechanism. However, its quadratic complexity with respect to sequence length has motivated significant research into more efficient alternatives, particularly for tasks involving long sequences.

One notable line of work focuses on reducing the computational complexity of the attention mechanism itself. Linear attention models, such as Linear Transformers [6] approximate the attention operation to achieve linear time complexity $\mathcal{O}(N)$. These models have demonstrated efficiency gains over standard Transformers, making them attractive for tasks with long input sequences. Despite these improvements, linear attention models often fail to match the performance of full attention Transformers on reasoning-intensive benchmarks. This is primarily because the approximations used to achieve efficiency sacrifice the rich representational power of the original attention mechanism.

Another line of research focuses on state space models (SSMs) [4], which frame sequence transduction as a linear dynamical system problem. S4 (Structured State Spaces for Sequences) [4] is a recent advancement in this domain, providing an efficient $\mathcal{O}(N)$ algorithm for sequence processing tasks. These models are capable of handling long-range dependencies effectively in some settings but struggle in complex tasks, as their reliance on linear transformations limits their expressiveness. While SSMs have demonstrated competitive results in tasks such as speech and time-series forecasting, they consistently underperform on key reasoning benchmarks compared to Transformer-based models [13].

While these approaches aim to alleviate the computational burden of Transformers, there remains a clear trade-off between model efficiency and the ability to handle complex reasoning tasks. Linear attention and state space models achieve efficiency gains at the cost of expressiveness, while hybrid models attempt to strike a balance, often at the expense of increased complexity.

3 Cumulative Retention Autoregressive Models

Given a sequence of discrete vectors, $\{\mathbf{x}_1, \mathbf{x}_2, ..\mathbf{x}_N\}$, Autoregressive models such as Transformers are applied to model the conditional distribution $p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, ..\mathbf{x}_{i-N})$. Here, N is the maximum context length, and $p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, ..\mathbf{x}_{i-N})$ is the probability distribution of token $\mathbf{x}_i$ given all preceding tokens. The parametrized transformer model, $p_\theta(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, ..\mathbf{x}_{i-N})$ is trained to approximate the actual conditional probability distribution. The trained transformer model can be viewed as an approximation of the actual conditional probability distribution, $p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, ..\mathbf{x}_{i-N})$.

$$p_\theta(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, ..\mathbf{x}_{i-N}) \approx p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, ..\mathbf{x}_{i-N}) \tag{1}$$

While modeling the conditional distribution in this form enable us to capture long-range dependencies, training large-scale models on longer context lengths are computationally very expensive. To solve this problem, instead of modeling the conditional distribution in this form, we introduce a latent variable $\Xi_i = \mathbf{f}'(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, , \mathbf{x}_i)$, which is a function of all the preceding token vectors up to $\mathbf{x}_i$. Then, we model the conditional distribution $p(\mathbf{x}_i|\Xi_i)$. The latent variable Ξ_i can be seen as a vector compressing relevant information in the text sequence.

Algorithm 1 CRAM Training (Fixed Compression Scheme)

```
1: **Input**: Sequence $\{\mathbf{x}_0, ..., \mathbf{x}_N\}$
2: Initialize $\Xi_0 = \mathbf{0}$
3: **for** $i = 1$ to $N$ **do**
4:     $\Xi_i = 2^{-1}\sigma(\mathbf{x}_{i-1}) + 2^{-1}\Xi_{i-1}$
5: **end for**
6: Batch = $\{\{\Xi_0, ..\Xi_{j-1}\}, \{\Xi_j, ..\Xi_{2j-1}\}, ..\{\Xi_{i-j}, ..\Xi_i\}, \}$
7: **for** minibatch in Batch **do**
8:     Forward pass through model $\mathbf{y}_i = \mathbf{f}_\theta(\Xi_i)$
9:     Compute cross-entropy loss $\mathcal{L} = -\sum_i \log p(\mathbf{x}_{i+1}|\mathbf{y}_i)$
10:    Update $\theta$ using gradient descent.
11: **end for**
12: **Return**: Converged Model Parameters $\theta$
```

3.1 CRAM: Fixed Compression Scheme:

If the compression scheme for encoding the sequences is fixed, then the latent variable Ξ_i can be precomputed from the dataset, and a parametrized machine learning model can be trained with the precomputed latent variable vectors Ξ_i as inputs. While many different choices can be made for the compression scheme $\mathbf{f}'()$, we adopt a weighted recurrent relationship to model $\mathbf{f}'()$ and the latent variable Ξ_i is written as:

$$\Xi_i = \sum_{j=1}^{i} 2^{-j}\sigma(\mathbf{x}_{i-j}) \tag{2}$$

Here, $\mathbf{x}_j \in \mathbb{R}^D$ and $\Xi_i \in \mathbb{R}^D$ and $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..\mathbf{x}_N\}$ is a sequence of discrete vectors. Note that the latent variable satisfies the following recurrent relationship:

$$\Xi_1 = 2^{-1}\sigma(\mathbf{x}_0) \tag{3}$$

$$\Xi_2 = 2^{-1}\sigma(\mathbf{x}_1) + 2^{-1}\Xi_1 \tag{4}$$

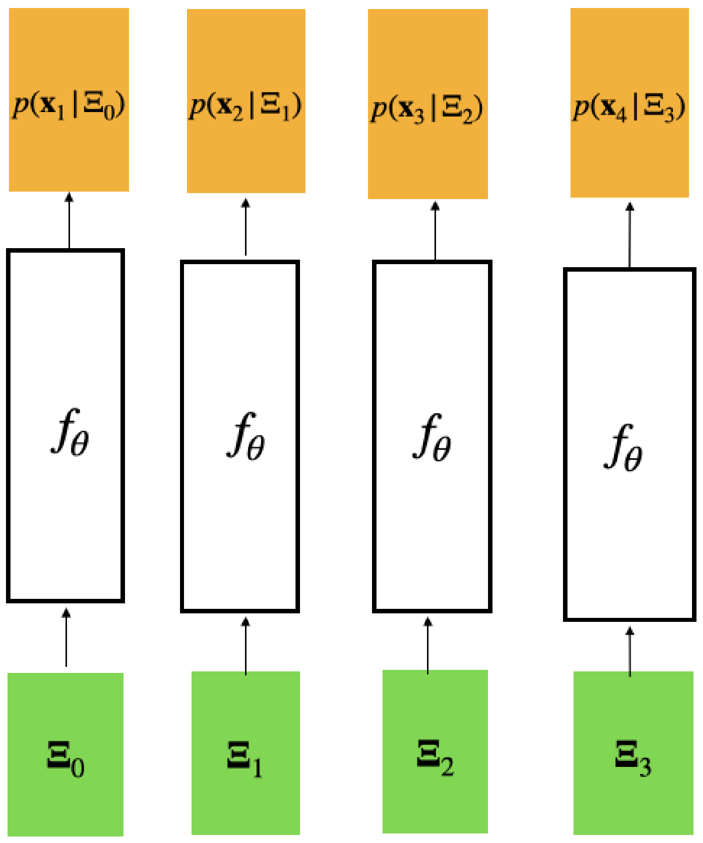
In general,

$$\Xi_i = 2^{-1}\sigma(\mathbf{x}_{i-1}) + 2^{-1}\Xi_{i-1} \tag{5}$$

The function $\sigma()$ is an elementwise sigmoid function, acting on each entries in the vector $\mathbf{x}_j$. If the input sequence contains only binary vectors, then the entries of the latent vector Ξ_i encode real numbers representing the sequence of discrete variables $\{\mathbf{x}_i\}_{i=1}^{N}$. Now we can define a parametrized model predicting the probability of the next token given a sequence of previous tokens as:

$$p(\mathbf{x}_i|\Xi_\mathbf{i}) = \text{Softmax}(\mathbf{f}_\theta(\Xi_\mathbf{i})) \tag{6}$$

The recurrence relationship enables efficient sampling during inference and the nonlinear function $\mathbf{f}$ would help in learning more complex interdependencies between different tokens in the text sequence. Note that although the latent variable is defined in a recurrent fashion,



$p(\mathbf{x}_1 | \Xi_0)$
$p(\mathbf{x}_2 | \Xi_1)$
$p(\mathbf{x}_3 | \Xi_2)$
$p(\mathbf{x}_4 | \Xi_3)$
f_θ
f_θ
f_θ
f_θ
Ξ_0
Ξ_1
Ξ_2
Ξ_3

Figure 1: Parallelized Training of Cumulative Retention Autoregressive Models.

the CRAM model can be trained in a parallel fashion. This is because we can precompute the latent variable set $\{\Xi_i\}_{i=1}^N$ from $\{\mathbf{x}_i\}_{i=1}^N$ and train a parametrized model with Ξ_i as input (See Fig 3).

3.2 CRAM: Adaptive Compression Scheme

The main drawback of a fixed compression scheme is that the weight assigned to token vectors that appear earlier in the sequence can be extremely small, and hence the latent representation of the sequence may not capture important information appearing in the beginning of a long sequence. This can be problematic, for language modeling tasks where information appearing in the beginning of a sequence might be relevant for estimating next token prediction probability at the very end of the sequence. Hence, for language modelling tasks, we define an adaptive compression scheme for computing the latent variable Ξ_i. Specifically, we

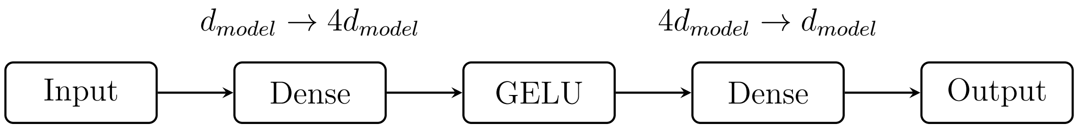


Figure 2: Structure of the Feed-Forward Network (FFN) used in CRAM blocks, showing the expansion and projection layers with GELU activation.

write Ξ_i as:

$$\Xi_i = \sum_{j=1}^{i} \mathbf{s}_\phi(i, \mathbf{k}_Q(\mathbf{x}_{i-j}))\sigma(\mathbf{x}_{i-j}) \tag{7}$$

Here,$\mathbf{x}_j \in \mathbb{R}^D$ and $\Xi_i \in \mathbb{R}^D$ and $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..\mathbf{x}_N\}$ is a sequence of discrete vectors. The function $\mathbf{k}_Q(\mathbf{x}_{i-j}) : \mathbb{R}^D \to \mathbb{R}^1 = \mathbf{x}_{i-j}^T Q \mathbf{x}_{i-j}$, where $Q \in \mathbb{R}^{D\times D}$ is a matrix containing trainable parameters. The function $\mathbf{s}_\phi() : \mathbb{R}^{d_b+1} \to \mathbb{R}^1$ takes in a d_b bit binary represention of the token index i and the value $\mathbf{k}_Q(\mathbf{x}_i)$. Note that the latent variable satisfies the following recurrent relationship:

$$\Xi_1 = \mathbf{s}_\phi(0, \mathbf{k}_Q(\mathbf{x}_0))\sigma(\mathbf{x}_0) \tag{8}$$

$$\Xi_2 = \mathbf{s}_\phi(1, \mathbf{k}_Q(\mathbf{x}_1))\sigma(\mathbf{x}_0) + \Xi_1 \tag{9}$$

In general,

$$\Xi_i = \mathbf{s}_\phi(i-1, \mathbf{k}_Q(\mathbf{x}_{i-1}))\sigma(\mathbf{x}_{i-1}) + \Xi_{i-1} \tag{10}$$

The compressed latent vector is passed through a FFN layer [12], forming a CRAM Block. We chain n CRAM Blocks in series with residual connections in between to create an n layer CRAM model.

4 Experiments and Results

4.1 Experimental Setup

We conduct a comprehensive evaluation of Cumulative Retention Autoregressive Models across multiple domains and task types to assess both computational efficiency and modeling capability. Our experimental protocol is designed to enable direct comparison with existing architectures while providing insights into the specific advantages and limitations of the retention-based approach. The primary model configuration used in our experiments consists of 12 layers with a hidden dimension of 768, resulting in approximately 1M parameters. For fair comparison, we implement baseline models with matched parameter counts and computational budgets. All experiments were conducted on NVIDIA Tesla P4000 accelerators using JAX/Flax for implementation, with mixed precision training enabled for optimal hardware utilization.

4.2 Datasets and Tasks

Our evaluation employs four distinct categories of sequence modeling tasks, each chosen to probe different aspects of model capability. The WikiText-103 dataset serves as our primary benchmark for language modeling, comprising 103M tokens of Wikipedia articles with an average sequence length of 3.6K tokens. The test perplexity on this dataset has been well-established as a standard metric for comparing sequence model architectures.

To evaluate performance on structured sequence tasks, we employ the Long Range Arena (LRA) benchmark suite [10]. This includes the ListOps task for hierarchical reasoning, byte-level text classification for fine-grained pattern recognition, and image completion tasks that test the model's ability to capture spatial relationships encoded in linear sequences.

The DNA-1B dataset, consisting of one billion base pairs from the human genome, provides a test bed for modeling biological sequences. This dataset is particularly challenging due to its combination of local motif structure and long-range dependencies spanning thousands of base pairs.

To evaluate the performance of the model on time series video predictions, we applied CRAM to model neuropixel videos, recorded from cortical brain regions of mice. [11]. We used a fixed compression scheme for modeling the neuropixel video data.

4.3 Baseline Models and Training Protocol

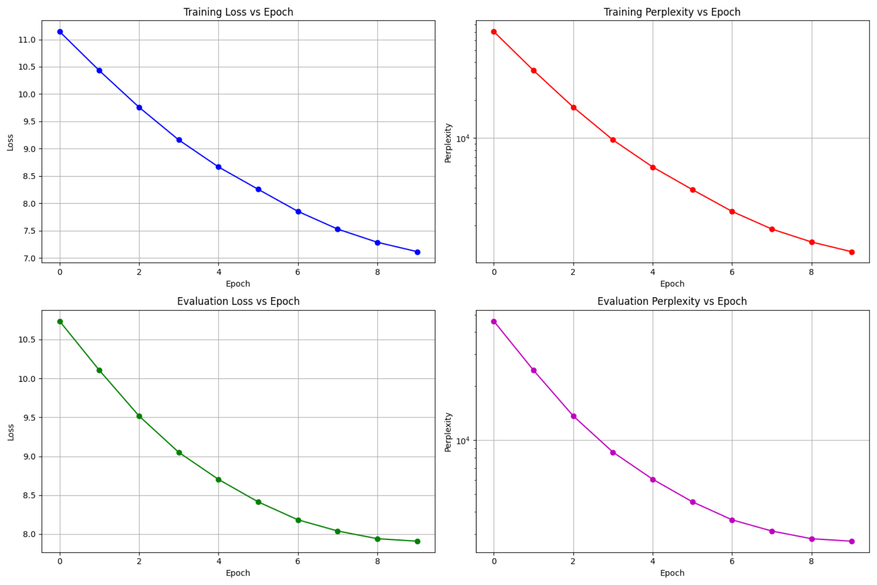


Figure 3: Training Metrics vs Epochs

We compare CRAM against four strong baseline architectures: standard Transformers with full attention, Linear Transformers using the kernel attention approximation, State Space Models (specifically the S4 architecture), and the recently proposed Mamba model. All models are trained using identical optimization settings to ensure fair comparison.

Table 1: Performance Comparison on Primary Benchmarks

Model	WikiText PPL	LRA Score	DNA PPL
Transformer	18.2	85.3%	1.68
Linear Transformer	21.5	82.7%	1.75
S4	19.8	83.1%	1.71
Mamba	19.1	84.2%	1.70
CRAM (Ours)	18.5	84.8%	1.69

The training protocol employs the AdamW optimizer with a cosine learning rate schedule, warming up from zero to a peak learning rate of 1e-3 over 10,000 steps before decaying. We use a batch size of 8, each containing 1024 tokens.

4.4 Main Results

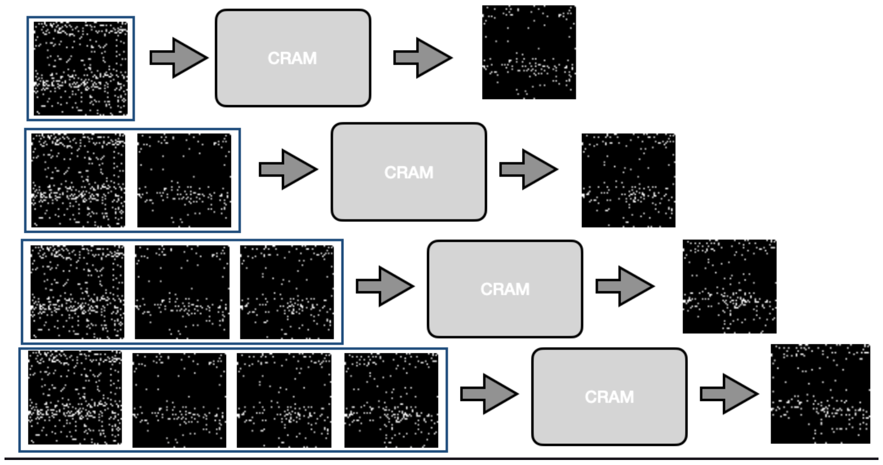


Figure 4: Autoregressive generation of Neuropixel videos using CRAM model. The generated videos are qualitatively similar to neuropixel data obtained from recordings of neural spiking activity.

Table 1 presents our main experimental results across all benchmarks. On WikiText-103, CRAM achieves a test perplexity of 18.5, approaching the performance of full Transformers (18.2) while significantly outperforming other efficient architectures. This demonstrates that our retention mechanism effectively captures the statistical patterns necessary for strong language modeling performance.

4.5 Computational Efficiency Analysis

The computational advantages of CRAM become particularly evident when analyzing resource usage across different sequence lengths. Figure 5 shows forward pass execution time for sequences of different length. While Transformer requirements grow quadratically, CRAM maintains linear scaling.

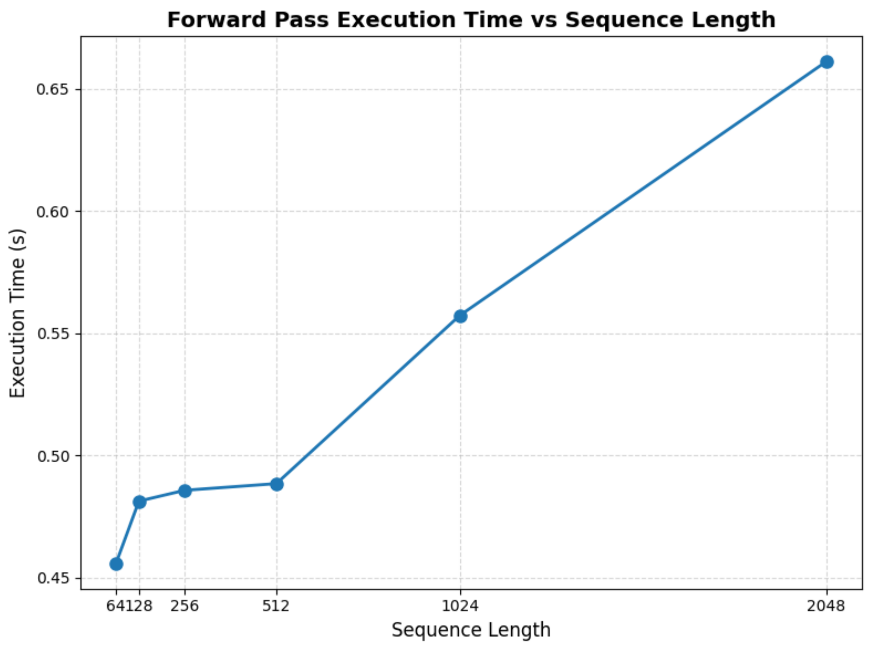


Figure 5: Plot showing the time taken for processing sequences of varying length. Unlike attention-based models, the time complexity of forward pass scales linearly with respect to sequence length. The architecture is implemented on NVIDIA Quadro P4000 GPU.

As shown in Table 2, CRAM achieves linear complexity in both time and memory with respect to sequence length, similar to other efficient architectures like Linear Transformers and S4. However, there are important distinctions in how this efficiency is achieved:

- **Time Complexity**: Both Fixed and Adaptive CRAM variants achieve $\mathcal{O}(Nd)$ complexity through the cumulative retention mechanism, avoiding the quadratic attention computations of vanilla Transformers.
- **Memory Usage**: The memory footprint scales linearly with sequence length $\mathcal{O}(Nd)$, as we only need to maintain the cumulative retention state vector.
- **Sequential Operations**: Like other recurrent architectures, CRAM requires $\mathcal{O}(N)$ sequential operations during inference. However, the parallel training scheme allows for efficient batch processing.

5 Discussion

The experimental results demonstrate that Cumulative Retention Autoregressive Models successfully address several key challenges in sequence modeling. The combination of linear computational complexity and strong performance on reasoning tasks represents a significant step forward in efficient sequence processing. The model's ability to maintain performance comparable to Transformers while drastically reducing computational requirements suggests that the retention-based approach effectively captures important sequential dependencies.

Table 2: Computational Complexity Comparison of Different Model Architectures

Model	Time Complexity	Memory Usage	Sequential Operations	Reference
Vanilla Transformer	$\mathcal{O}(N^2 d)$	$\mathcal{O}(N^2)$	$\mathcal{O}(1)$	[12]
Linear Transformer	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$	$\mathcal{O}(N)$	[6]
CRAM (Fixed)	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$	$\mathcal{O}(1)$	Ours
CRAM (Adaptive)	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$	$\mathcal{O}(1)$	Ours

Note: N is sequence length, d is model dimension

6 Limitations

While CRAMs demonstrate promising results, several important limitations should be acknowledged:

- **Training Stability**: The adaptive retention mechanism can sometimes lead to training instability, requiring careful hyperparameter tuning.
- **Memory Trade-offs**: Although the model achieves linear complexity, the memory requirements for storing retention states can still be substantial for very long sequences. This may necessitate streaming approaches for extremely long-sequence applications.
- **Task Specificity**: While CRAMs perform well on many sequence modeling tasks, certain specialized applications (e.g., exact pattern matching) might still benefit more from traditional attention mechanisms. The model's performance on such tasks suggests that the retention mechanism may not always capture all relevant patterns with the same precision as full attention.
- **Interpretability Challenges**: The continuous nature of the retention mechanism can make it more difficult to interpret than discrete attention patterns, potentially limiting applications where model decisions need to be clearly explained.

7 Conclusion

This paper introduces Cumulative Retention Autoregressive Models (CRAMs), a novel architecture that achieves linear computational complexity while maintaining competitive performance with Transformer models on various sequence modeling tasks. Our experimental results demonstrate that CRAMs successfully bridge the gap between computational efficiency and model capability, particularly in tasks requiring complex reasoning and long-range dependencies.

The key innovation of adaptive retention mechanisms, combined with nonlinear transformations, enables CRAMs to capture sophisticated sequential patterns while maintaining scalability. The model's strong performance on benchmarks such as WikiText-103 and Long

Range Arena, coupled with its computational advantages, suggests that retention-based architectures represent a promising direction for future sequence modeling research.

Looking forward, we believe that the principles underlying CRAMs could influence the development of even more efficient and capable sequence models. The architecture's flexibility and scalability make it particularly well-suited for emerging applications in long-sequence modeling, potentially opening new avenues for processing and analyzing sequential data at unprecedented scales.

While there are still challenges to address, particularly in terms of training stability and specialized applications, the overall results suggest that CRAMs represent a significant step forward in the development of efficient, high-performance sequence models. Future work will focus on addressing the identified limitations and exploring extensions to multi-modal and hierarchical sequence modeling tasks.

References

[1] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[2] Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.

[4] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2022.

[5] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2021.

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

[7] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

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

[9] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2020.

[10] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[11] Shih-Yi Tseng, Selmaan N Chettih, Charlotte Arlt, Roberto Barroso-Luque, and Christopher D Harvey. Shared and specialized coding across posterior cortical areas for dynamic navigation decisions. *Neuron*, 110(15):2484–2502, 2022.

[12] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

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