Optimizing Transformer Inference with Selective Distillation: Layerwise Conversion to Linear Attention

Yeonju Ro
yro@cs.utexas.edu
University of Texas at Austin
Austin, Texas, USA

Zhenyu Zhang
zhenyu.zhang@utexas.edu
University of Texas at Austin
Austin, Texas, USA

Vijay Chidambaram
vijayc@utexas.edu
University of Texas at Austin
Austin, Texas, USA

Aditya Akella
akella@cs.utexas.edu
University of Texas at Austin
Austin, Texas, USA

Abstract

Transformer-like architectures with softmax attention have demonstrated exceptional performance in language modeling by capturing long-term token dependencies and ensuring efficient training. However, their scalability is constrained by the quadratic computation costs incurred during inference. To mitigate this issue, subquadratic models with SSMs and linear attentions have been introduced; however, such models achieve lower accuracy than Transformers. In this work, we aim to get the best of both worlds: we seek to convert carefully selected attention layers in a Transformer to gated linear attention layers, without sacrificing accuracy. Specifically, we analyze the performance benefits of subquadratic models and propose a distillation method that progressively converts the attention layer to a gated linear attention layer [25]. While selecting layers to convert, we leverage downstream task accuracy as a criterion to minimize the accuracy loss caused by conversion. Our evaluation demonstrates that task accuracy is an effective criterion that can be used in adaptive distillation.

1 Introduction

With the success of language modeling with transformer-based architecture, many transformer-based large language models (LLMs) have been released and deployed for various tasks such as code generation [16], translation [7], question-answering [5, 20], and text summarization [11]. The superior performance of transformer architecture compared to alternative architectures (e.g., RNN or LSTM) is rooted in the attention layer, which calculates the relevance or importance of each token to another and is known as the attention score. By directly calculating the attention score for each pair of tokens, transformer-based architecture can better capture global context and long-range dependencies over long sequences without suffering from issues like vanishing or exploding gradient problems. Besides the attention layer's efficacy in language modeling compared to its alternatives, it is amenable to parallelization by processing all tokens of inputs simultaneously, leading to significantly fast training.

Yet, the scalability of the attention layer remains a challenge, as the computation complexity increases quadratically to the sequence length (i.e., the number of tokens in a sequence). This computation cost becomes a burden, especially in the inference, where parallelization is unfeasible due to the iterative generation in the decoding phase. Unlike training where all tokens are provided at once, only one token is generated and processed at a time during the decoding or generation steps, blocking parallel computation. Moreover, the runtime memory for storing the intermediate states (e.g., key and value vectors in attention mechanism) known as KV Cache [13] also grows proportionally to the sequence length and takes a significant portion of runtime memory [6].

To mitigate the scalability problems during inference, recent work studied the linearization of the attention layer by approximating the softmax function with learned kernel functions [4, 19], replacing the attention layer with RNN or SSM-based layers which have linear time complexity to the sequence length and maintain constant intermediate states, or using mixed architecture of transformer and RNN or SSM [9, 22]. In particular, distillation-based methods [1, 18, 23, 26] have gained attention for leveraging the advantages of transformers during training and linear attention during inference. While early works showed decent accuracy in fully linearized models, knowledge distillation as an optimization method often results in some inevitable loss of accuracy.

In this work, we build upon the principles of distillation-based approaches and introduce a disciplined framework for knowledge distillation. Our framework progressively transfers the representations of the quadratic attention layers in pre-trained transformer models to linear attention layers, based on each layer's sensitivity to distillation. Specifically, we evaluate the impact of each layer on the model's performance and use this metric to selectively convert layers. By adjusting the conversion rate, our framework offers flexibility in balancing inference efficiency with model performance.

The 2nd Workshop on Hot Topics in System Infrastructure (HotInfra'24), co-located with SOSP'24, November 3, 2024, Austin, TX, USA

It also supports service-level objectives (SLOs) for latency by automatically determining the conversion rate when a target latency is given.

2 Method

2.1 Overview

In this work, we propose a framework for systematically selecting which layers to convert from self-attention to linear attention. Unlike prior approaches [1, 18, 23, 26] that convert all layers to linear attention to maximize performance benefits, our framework offers flexibility by adaptively adjusting the conversion rate to balance system performance and accuracy. We utilize a gated linear attention architecture [25] for this conversion, which incorporates data-dependent features similar to selective SSMs (e.g., Mamba [23]) and is known for improved throughput.

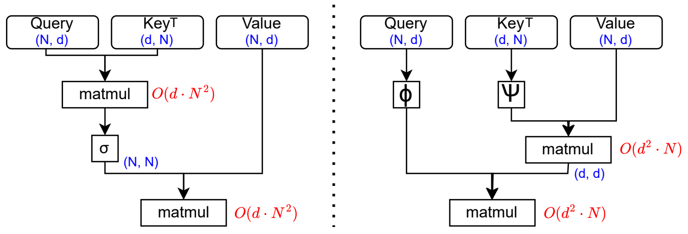

Figure 1. By approximating softmax function σ with ϕ and ψ, we can calculate self-attention in order of $\phi(q) \cdot (\psi(K)^T \cdot V)$ by making use of the associative property of matrix multiplication, reducing the computation complexity from $O(N^2 \cdot d)$ to $O(d^2 \cdot N)$. Blue text indicates the shape of the tensor and red text indicates the computational complexity of each matrix multiplication. N is sequence length and d is hidden size. While d is deterministic for a given model size, N scales based on the user's input.

For a given pre-trained transformer model with self-attention, we first train kernel functions ϕ and ψ to approximate the softmax operation, such that $softmax(qK^T) \approx \phi(q)\psi(K)^T$ [2, 4]. This approximation allows the computation of $K \times V$ to be performed before $q \times K$ due to the associative property of matrix multiplication, thereby reducing the computational cost from $O(N^2 \cdot d)$ to $O(d^2 \cdot N)$ (Figure 1). We use knowledge distillation to train ϕ and ψ, as detailed in §2.4, rather than training from scratch.

The learned kernel functions enable the conversion of each transformer layer into a gated linear attention layer. Our conversion approach is layer-wise, allowing us to choose between a transformer layer and a linear attention layer for each layer. Based on the selection criterion analysis discussed in §2.2 and target service-level objective discussed in §2.3, we select n layers out of N total layers to convert to linear attention, resulting in n linear attention layers and $N - n$ transformer layers.

2.2 Analyzing Selection Criterion

In this section, we introduce various criteria for layer selection. Specifically, we explore three options: layer position, attention distribution, and layerwise sensitivity to downstream tasks. We discuss how each criterion influences the choice of model architectures and their impact on overall performance.

Layer Position: Early and late layers in a model often aggregate global information and might benefit from the sequential information captured by RNNs (e.g., linear attention). In contrast, middle layers typically focus on detailed context by computing pairwise relationships between tokens, where self-attention mechanisms are more effective [4, 8].

Attention Distribution: Analyzing attention weights across layers can help identify which layers have more focused and meaningful attention distributions. Layers with less focused attention might benefit more from RNN-based mechanisms that excel in capturing sequential dependencies [14, 26].

Layerwise Sensitivity: Our conversion approach operates on a per-layer basis, allowing us to measure sensitivity by replacing one layer with the converted layer at a time. This sensitivity can be assessed using downstream task accuracy, providing a more direct measure of how each layer responds to the conversion.

In our evaluation (§3.3), we compare the performance of each criterion when applied during the conversion process.

2.3 Determining Conversion Rate

Serving systems often operate under specific service-level objectives (SLOs). In this work, we use inference latency and peak memory as the primary SLO. Since the computations in transformer and gated linear layers are deterministic and determined by the sequence length, the latency and memory can be efficiently interpolated by performing a single inference run at each end of the conversion spectrum. This approach enables us to identify the optimal conversion rate that satisfies the target latency or memory budget. Given a latency budget lat_b, we measure the latency lat_0 using a base transformer model and lat_1 using a fully linear model. Note that for measuring lat_1, exact model parameters are not required; random parameters can be used solely for latency measurement. The maximum conversion rate r can then be obtained using the following equation (Equation 1). Similarly, a conversion rate can be determined for a given memory budget.

$$r = \arg\max \{ r \mid \text{lat}_b \geq |r \cdot \text{lat}_0 + (1 - r) \cdot \text{lat}_1| \} \tag{1}$$

2.4 Distilling Self-attention to Linear Attention

Our distillation process operates on a per-layer basis, where we train kernel functions by matching the input and output of each layer. We utilize 512 sequences from the C4 training dataset, collecting queries, keys, values, and outputs from each layer.

We then align the query vector q and $\phi(q)$, as well as the key vector k and ψ(k). For ψ and ϕ, we use multi-layer perceptron (MLP) and GELU following the spirit of prior work [2, 26] (Eq. 2). The global objective is to minimize the l2 distance between the attention output, $attn\_out$, and the output of the gated linear attention, out. The combined loss function is shown in Eq 3.

$$\begin{aligned}\phi(q) &= \left|\sigma_\phi\left(\sigma_\phi(qW_{\phi,1})W_{\phi,2}\right)\right| \\ \psi(k) &= \left|\sigma_\psi\left(\sigma_\psi(kW_{\psi,1})W_{\psi,2}\right)W_{\psi,3}\right|\end{aligned} \tag{2}$$

$$\mathcal{L} = \|\text{attn\_out} - \text{out}\| + \lambda_1 \cdot \|q - \phi(q)\| + \lambda_2 \cdot \|k - \psi(k)\| \tag{3}$$

For training, we employ the Adam optimizer. We only train parameters in ϕ and ψ for 40 epochs and do not change the model parameters. The learning rate is initialized at 0.0001 and follows a cosine annealing schedule. Note that the training is done offline and one-time cost, and no cost is added during the inference. (λ_1, λ_2) is hyper-parameter and set to $(1e-4, 1e-2)$

Once converted, the linear attention mechanism can compute the attention layer in a recursive manner, similar to RNNs. We employed the Gated Linear Attention architecture [25], which incorporates a data-dependent gate that selectively weighs hidden states based on the input. At time step t, the hidden state S_t is updated by applying the Hadamard product between the data-dependent gate $G_t \in (0,1)^{(d\times d)}$ (for hidden size d) and previous hidden states S_{t-1}, along with the outer product of the transposed key vector k_t^T and value vector v_t. Output of attention layer o_t is achieved by multiplying query vector q_t to the hidden states (Eq. 4).

$$\begin{aligned}S_t &= G_t \odot S_{t-1} + \psi(k_t)^T v_t \\ o_t &= \phi(q_t) \cdot S_t\end{aligned} \tag{4}$$

3 Evaluation

3.1 Experimental Setup

In our experiments, we used a single A100 GPU with 80GB of memory. For the base transformer model, we used Llama2-7b model [21] and distilled it into gated-linear-attention [25]. For distillation, we used 512 sequences of the C4 training dataset [15], and the hidden layer size of the learned kernel is set to 512. To evaluate the model accuracy, we used long-text summarization tasks on two datasets (CNN/Dailymail Dataset [3, 17], Xsum Dataset [12]) and measured ROUGE score [10], which measures the overlap of unigrams (individual words) between a generated text and a reference text.

3.2 System Performance

In this section, we measure the benefits of conversion from transformer to linear attention. Figure 2 shows the inference latency and peak memory as a function of conversion rate. Latency is measured in an isolated environment while generating 1024 tokens given a 1024-token prompt after 20 warm-up generations. Memory includes space for inputs, intermediate states, and model parameters.

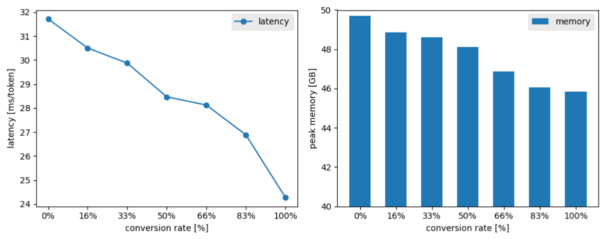

Figure 2. Figures show latency (left) and peak memory usage (right) by conversion rate. 0% shows base transformer model and 100% shows the fully linear attention model. Both metrics show a linear decrease with increasing layer conversion, suggesting that this can serve as a primary method for determining the conversion rate for given system constraints.

The results demonstrate that inference latency and memory usage have linear relationship with the conversion rate, suggesting that this method could provide a cost-effective way to determine the optimal rate. Given a latency or memory budget, our framework selects $N \times conversion\_rate$ layers based on the criteria outlined in Section §2.2, and loads them into memory, ensuring no additional cost is incurred during inference.

3.3 Efficacy of Conversion Criteria

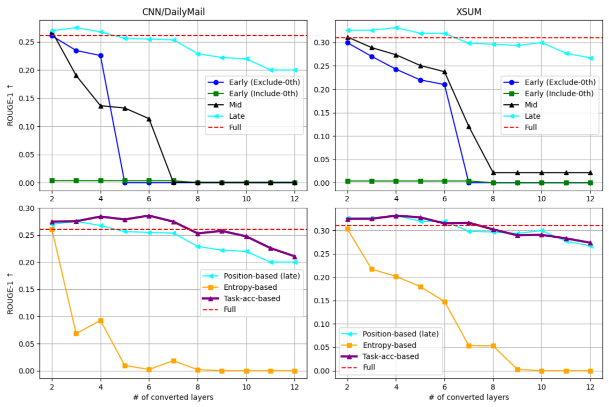

Figure 3. Figure shows ROUGE-1 scores for long document summarization tasks across CNN/DailyMail and XSUM datasets with different selection criteria. The upper row compares the efficacy of selecting early (exclude-0th), early (include-0th), mid, and late layers, while the lower row compares position-based (late), entropy-based, and task-accuracy-based selections. Higher y-axis values indicate better performance. The red dotted line plots the full accuracy achieved by the base transformer model.

In this section, we evaluate the various selection criteria outlined in §2.2. First, we examine the effectiveness of layer-position-based conversion by categorizing the layers into three groups: (1) "early" layers (0-10), (2) "mid" layers (11-20), and (3) "late" layers (21-31). Within each group, we picked random layers to convert and assessed the performance of the converted model (Fig 3 (a)-(b)). Although previous studies suggest that RNNs could be effectively utilized in early and late layers [4, 8], our experiments reveal that randomly selecting layers for conversion can lead to significant performance degradation. In particular, converting certain layers (e.g., the 0-th layer) failed in performance.

To establish a disciplined selection process, we analyze the attention distribution and layer-wise sensitivity to the downstream task. Specifically, we compute the entropy of the attention scores [14] as a measure of attention distribution, and task accuracy as a measure of layer-wise sensitivity. The results are shown in Figure 3 (c)–(d), suggesting that task-accuracy-based layerwise sensitivity is more effective as a selection criterion. While entropy is convenient as it can be computed before kernel training, the results indicate that using entropy as a selection criterion is less effective than task-accuracy-based sensitivity. This is because certain layers are critical to performance when converted, and this stems from the training method. Therefore, it cannot be captured in the baseline model before training.Selecting random late layers consistently performs well, but this approach is not generalizable when more layers need to be selected after all late layers have been chosen (e.g., after selecting layers 21-31, what should be selected next?)

4 Conclusion and Future work

In this work, we present a distillation framework that converts a pre-trained transformer model into a gated linear attention model for more efficient inference. Unlike fully converting all layers into linear layers, our approach performs a layer-wise conversion, providing flexibility in balancing system performance and accuracy. This method also supports operation under a latency or memory budget, where the conversion rate is determined through interpolation. To minimize accuracy loss, our framework carefully selects which layers to convert based on their layer-wise sensitivity.

While this work adopted a distillation method that only trains kernels and gating layer instead of model parameters, a fine-tuning based approach [24] can also be used for conversion. We use latency as the primary target, other objectives such as aggregated accuracy or throughput could be considered as alternatives. Achieving these alternative goals may require a more in-depth analysis of resource usage and the development of feedback mechanisms to compensate for accuracy losses during conversion. We leave this exploration for future work.

References

[1] Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. 2024. Transformers to SSMs: Distilling Quadratic Knowledge to Subquadratic Models. arXiv:2408.10189 [cs.LG] https://arxiv.org/abs/2408.10189

[2] Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. 2024. Get More with LESS: Synthesizing Recurrence with KV Cache Compression for Efficient LLM Inference. *arXiv preprint arXiv:2402.09398* (2024).

[3] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems* 28 (2015).

[4] 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*. PMLR, 5156–5165.

[5] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[6] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[7] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model. (2023).

[8] Jiangyun Li, Sen Zha, Chen Chen, Meng Ding, Tianxiang Zhang, and Hong Yu. 2022. Attention guided global enhancement and local refinement network for semantic segmentation. *IEEE Transactions on Image Processing* 31 (2022), 3211–3223.

[9] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887* (2024).

[10] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[11] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[12] Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745* (2018).

[13] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems* 5 (2023), 606–624.

[14] Mélanie Prague and Marc Lavielle. 2022. SAMBA: A novel method for fast automatic model building in nonlinear mixed-effects models. *CPT: Pharmacometrics & Systems Pharmacology* 11, 2 (2022), 161–172.

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

[16] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[17] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint*

arXiv:1704.04368 (2017).

[18] Siavash Shams, Sukru Samet Dindar, Xilin Jiang, and Nima Mesgarani. 2024. Ssamba: Self-supervised audio representation learning with mamba state space model. *arXiv preprint arXiv:2405.11831* (2024).

[19] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 3531–3539.

[20] Yuchong Sun, Che Liu, Jinwen Huang, Ruihua Song, Fuzheng Zhang, Di Zhang, Zhongyuan Wang, and Kun Gai. 2023. Parrot: Enhancing multi-turn chat models by learning to ask questions. *arXiv preprint arXiv:2310.07301* (2023).

[21] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[22] Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. 2024. An Empirical Study of Mamba-based Language Models. *arXiv preprint arXiv:2406.07887* (2024).

[23] Junxiong Wang, Daniele Paliotta, Avner May, Alexander M Rush, and Tri Dao. 2024. The Mamba in the Llama: Distilling and Accelerating Hybrid Models. *arXiv preprint arXiv:2408.15237* (2024).

[24] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. 2016. Network morphism. In *International conference on machine learning*. PMLR, 564–572.

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

[26] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. 2024. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347* (2024).