When Linear Attention Meets Autoregressive Decoding: Towards More Effective and Efficient Linearized Large Language Models

Haoran You [1] **Yichao Fu** [1] **Zheng Wang** [1] **Amir Yazdanbakhsh** [2] **Yingyan (Celine) Lin** [1]

Abstract

Autoregressive Large Language Models (LLMs) have achieved impressive performance in language tasks but face two significant bottlenecks: (1) quadratic complexity in the attention module as the number of tokens increases, and (2) limited efficiency due to the sequential processing nature of autoregressive LLMs during generation. While linear attention and speculative decoding offer potential solutions, their applicability and synergistic potential for enhancing autoregressive LLMs remain uncertain. We conduct the first comprehensive study on the efficacy of existing linear attention methods for autoregressive LLMs, integrating them with speculative decoding. We introduce an augmentation technique for linear attention that ensures compatibility with speculative decoding, enabling more efficient training and serving of LLMs. Extensive experiments and ablation studies involving seven existing linear attention models and five encoder/decoder-based LLMs consistently validate the effectiveness of our augmented linearized LLMs. Notably, our approach achieves up to a 6.67 reduction in perplexity on the LLaMA model and up to a 2× speedup during generation compared to prior linear attention methods. Codes and models are available at https://github.com/GATECH-EIC/Linearized-LLM.

1. Introduction

LLMs have demonstrated exceptional capabilities in language understanding and generation tasks, sparking immense interest. Autoregressive LLMs, like OpenAI's ChatGPT (OpenAI, 2023a;b), Meta's LLaMA (Touvron et al., 2023a;b), and Google's Gemini (Anil et al., 2023), have achieved state-of-the-art (SOTA) performance in generation. However, these models suffer from significant computational and memory demands, hindering their efficiency in both training and serving. These limitations stem from two key bottlenecks: ***Bottleneck 1***: The attention module, a core component of LLMs, exhibits quadratic complexity relative to the input sequence length. This necessitates training LLMs with limited context sizes (e.g., 2048 tokens for LLaMA), restricting their ability to process lengthy documents or engage in extended conversations (Chen et al., 2023c). ***Bottleneck 2***: The sequential nature of autoregressive decoding limits parallelism during generation, resulting in slow inference speeds, especially for long sequences (Miao et al., 2023).

Various techniques have been proposed to address these bottlenecks, including pruning (Ma et al., 2023), quantization (Frantar et al., 2022; Xiao et al., 2023; Harma et al., 2024), speculative decoding (Miao et al., 2023; Leviathan et al., 2023), and linear attention (Qin et al., 2023; Lu et al., 2021). Among these, linear attention tackles ***Bottleneck 1*** by reducing the quadratic complexity of softmax attention from quadratic to linear. Speculative decoding addresses ***Bottleneck 2*** by employing smaller draft models for speculative parallel generation, followed by verification using the full LLM (Miao et al., 2023; Cai et al., 2023b; Chen et al., 2023a). While promising, the effectiveness of these techniques, especially when combined with autoregressive LLMs, remains largely unexplored. This paper addresses two critical questions: ***Q1***: Can existing linear attention methods, primarily designed for encoder-based LLMs like BERT (Devlin et al., 2018) or Vision Transformers (ViTs) (Dosovitskiy et al., 2021), be effectively applied to autoregressive decoder-based LLMs? ***Q2***: Can linear attention and speculative decoding be seamlessly integrated to address both bottlenecks concurrently during LLM training and serving?

We conduct the first comprehensive empirical exploration to evaluate the efficacy of linearized autoregressive LLMs and their compatibility with speculative decoding. Our findings for ***Q1*** reveal that directly applying existing linear attention methods to autoregressive LLMs leads to suboptimal performance, due to the disruption of temporal dependencies cru-

[1]School of Computer Science, Georgia Institute of Technology, Atlanta, USA [2]Google DeepMind, Mountain View, USA. Correspondence to: Haoran You <haoran.you@gatech.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

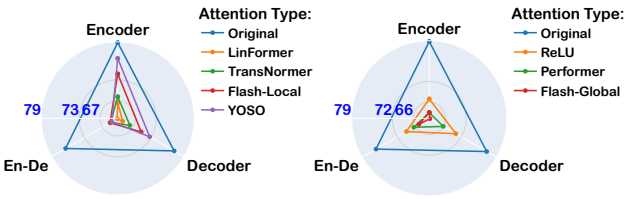

Figure 1. Empirical evaluation of seven linear attention methods on top of three types of LLMs on the GLUE (Wang et al., 2018) benchmark: (1) encoder-based BERT (Devlin et al., 2018); (2) decoder-based GPT-2 (Radford et al., 2019); and (3) encoder-decoder T5 (Roberts et al., 2022). **Left:** The majority of SOTA linear attentions, including LinFormer (Wang et al., 2020), TransNormer(Qin et al., 2022), FLASH-Local (Hua et al., 2022), and YOSO (Zeng et al., 2021), exhibit superior performance on encoder-based models compared to decoder-based ones. **Right:** Other linear attention methods, such as ReLU-based one (Cai et al., 2023a), Performer (Choromanski et al., 2021), and FLASH-Global (Hua et al., 2022), consistently perform less effectively on all LLMs.

cial for autoregressive generation. For instance, convolution-based augmentation techniques (You et al., 2023b; Xiong et al., 2021) introduce "information leakage" from future tokens during training, i.e., they use convoluted future context directly instead of predicting the next tokens. Addressing ***Q2***, we find that direct integration of linear attention with speculative decoding is ineffective, owing to mismatches in handling temporal dependencies. In particular, speculative decoding employs "tree-based" attention, complicating the application of standard linear attention methods. Motivated by these challenges, we propose an effective local convolutional augmentation to prevent information leakage, boost performance, and maintain compatibility with speculative decoding. Our key contributions are:

- We conduct a comprehensive evaluation of seven linear attention methods across three types of LLMs (encoder-based, decoder-based, and encoder-decoder), revealing that existing encoder-based linear attentions are not optimally suited for autoregressive decoder-based LLMs.
- We introduce an ***effective*** local augmentation technique that enhances the local feature extraction capabilities of linear attention in autoregressive LLMs while preventing information leakage.
- We develop a solution for seamlessly integrating linear attention with speculative decoding's tree-based attention, boosting token-level parallelism for ***efficient*** generation and accelerating both LLM training and serving.
- Extensive experiments on five LLMs validate the effectiveness of our augmented linearized LLMs, achieving up to a 6.67 reduction in perplexity and up to 2× speedups during generation over existing linear attention methods.

2. Related Works

Autoregressive LLMs. Existing LLMs are broadly categorized into three architectures: *encoder-based*, *decoder-based*, and *encoder-decoder* models. Encoder-based models like BERT (Devlin et al., 2018) focus on natural language understanding and are also commonly used in image processing (Dosovitskiy et al., 2021). Encoder-decoder models, such as Transformer (Vaswani et al., 2017), are designed for sequence-to-sequence tasks, where the encoder extracts features and the decoder generates outputs. Decoder-based models, including GPT (Radford et al., 2019; OpenAI, 2023b) and LLaMA (Touvron et al., 2023a), generate text sequentially by predicting the next token. While all these models utilize Transformer architectures, their specific design and purpose vary. This paper presents a comprehensive study of applying linear attention techniques to both encoder-decoder and decoder-based LLMs.

Efficient Linear Attention Self-attention in transformers, with their quadratic computational complexity (Zhu et al., 2021; Katharopoulos et al., 2020), have led to the development of linear attention methods. Kernel-based linear attentions (Liu et al., 2021; Arar et al., 2022; Wang et al., 2020; Tu et al., 2022) decompose the softmax with kernel functions and change the computation order. However, few approaches focus on decoder-based autoregressive LLMs (Hua et al., 2022; Katharopoulos et al., 2020). Recent studies, such as LongLoRA (Chen et al., 2023c), aim to adapt local attention techniques for efficient fine-tuning, but a thorough comparison of linear attention methods for autoregressive LLMs is less explored. This paper systematically review existing linear attention for decoder-based autoregressive LLMs and investigates how to efficiently enhance less effective linear attention methods.

Speculative Decoding. Linear attention methods reduce training inefficiencies, but the sequential nature of autoregressive decoding limits parallelism during deployment, restricting the number of input tokens. Speculative decoding (Chen et al., 2023a; Miao et al., 2023; Kim et al., 2023; Leviathan et al., 2023; Cai et al., 2023b) has proven to be an effective strategy for boosting parallelism in LLM serving. It utilizes small speculative models for initial generation, with the original LLMs validating the outputs. Recent works, such as Medusa (Cai et al., 2023b), suggests that these models can be the same. This paper investigates the synergy between linearized LLMs and speculative sampling to improve LLM training and serving efficiency.

Additional related works are provided in Appendix A.

3. Preliminaries and Evaluation

Self-Attention and Softmax Bottleneck. The self-attention module is a core component of the Transformer (Vaswani

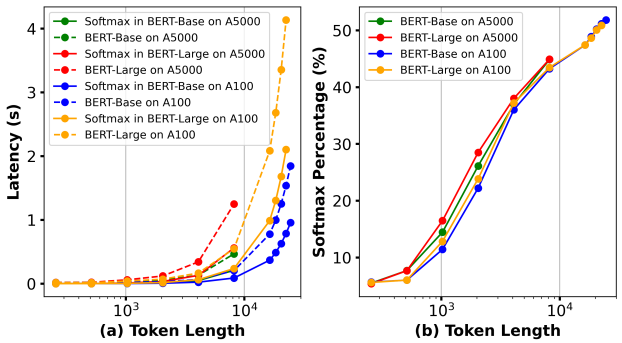

Figure 2. Runtime profiling: **(a)** actual runtime latencies for both the softmax and the entire model; **(b)** the percentage of time allocated to softmax computations across the latency of the entire model. All data were collected using BERT-Base/Large models on a single A5000 or A100 GPU.

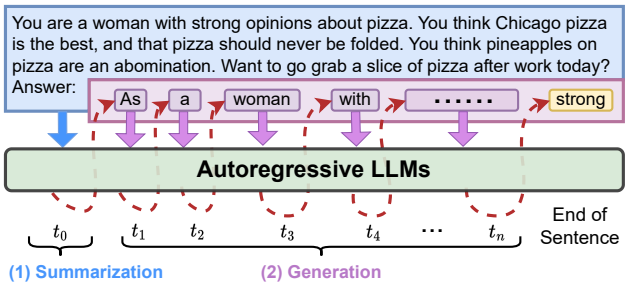

Figure 3. Illustrating the autoregressive LLMs. The process of generating text unfolds in two stages: (1) an initial *summarization or prefill phase* that employs a large batch size and utilizes the given input context; followed by (2) the *generation or decode phase*, which operates on a single-batch basis, using previously generated tokens to continue the text output.

et al., 2017; Dosovitskiy et al., 2021), and typically includes multiple heads. Each head computes global-context information by evaluating pairwise correlations among all n tokens (n represents the total number of tokens) as follows:

$$\begin{aligned} \texttt{Attn}(\mathbf{X}) &= \texttt{Concat}(\mathrm{H}_1, \cdots, \mathrm{H}_h) \cdot \mathbf{W}_O, \text{ where} \\ \mathrm{H}_i &= \texttt{Softmax}\left(\frac{f_Q(\mathbf{X}) \cdot f_K(\mathbf{X})^T}{\sqrt{d_k}}\right) \cdot f_V(\mathbf{X}), \end{aligned} \tag{1}$$

where h denotes the number of heads. Within each head H_i, input tokens $\mathbf{X} \in \mathbb{R}^{n \times d}$ of length n and dimension d will be linearly projected to query, key, and value matrices, i.e., $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d_k}$, through three linear mapping functions, $f_Q = \mathbf{X}\mathbf{W}_Q, f_K = \mathbf{X}\mathbf{W}_K, f_V = \mathbf{X}\mathbf{W}_V$, where $d_k = d/h$ is the dimensionality of each head and $\mathbf{W}_{Q/K/V}$ are the associated weight matrices. The final outputs are generated by concatenating the results from all heads and applying a weight matrix $\mathbf{W}_O \in \mathbb{R}^{d \times d}$.

Attention is the bottleneck in LLMs, accounting for 55% of the total runtime during autoregressive generation (Appendix C). Within self-attention, softmax becomes a memory bottleneck when handling long sequences (Dao et al., 2022; Kao et al., 2023). As depicted in Fig. 2, we profiled BERT-Base/Large models on a single A100/A5000 GPU to illustrate the percentage of time allocated to softmax as the token length increases. We observe that the runtime percentage for softmax continues to increase *quadratically* as the token length grows, occupying *up to 50%* of the total model latency when token length reaches 10^4.

Linear Attentions (LAs). Kernel-based LAs (Katharopoulos et al., 2020; Wang et al., 2020; You et al., 2023b) have emerged as an effective method for eliminating the need for softmax and reducing the quadratic complexity. The core idea is to decompose the similarity measurement function, typically based on softmax, into separate kernel embeddings, i.e., $\mathrm{Sim}(\mathbf{Q}, \mathbf{K}) \approx \phi(\mathbf{Q}) \cdot \phi(\mathbf{K})^T$. This enables rearranging the computation order to $\phi(\mathbf{Q})(\phi(\mathbf{K})^T\mathbf{V})$, utilizing the associative property of matrix multiplication. Consequently, the complexity of attention becomes linear relative to the feature dimension d, instead of quadratic with respect to the token length n. These LAs, however, could lead to a significant accuracy drop compared to softmax-based attention unless they are carefully designed.

Autoregressive LLMs. As depicted in Fig. 3, unlike the initial summarization phase, which processes a large number of tokens simultaneously and is computationally intensive, the generation phase faces severe memory or bandwidth limitations due to its autoregressive nature, involving token-by-token generation. Linear attention speeds up training and reduces summarization complexity, but is less effective for autoregressive generation due to low parallelism. Speculative decoding emerges as a critical method for increasing parallelism. Thus, ensuring compatibility between linear attention and speculative decoding is imperative for efficient summarization and generation phases.

3.1. Evaluation of Existing LAs on LLMs

Comprehensive Evaluation. To investigate whether previous LAs can be generally applicable to three categories of LLMs: encoder-based, decoder-based, and encoder-decoder, we evaluate seven distinct LAs, including FLASH-Local&Gloabl (Hua et al., 2022), Linformer (Wang et al., 2020), Performer (Choromanski et al., 2021), TransNormer (Qin et al., 2022), YOSO (Zeng et al., 2021), ReLU (Cai et al., 2023a), across three representative LLMs in each category: encoder-based BERT (Devlin et al., 2018), decoder-based GPT-2 (Radford et al., 2019), and encoder-decoder T5 (Raffel et al., 2020). As detailed

Table 1. Evaluation of seven LAs on BERT (Devlin et al., 2018), an encoder-based LLM, with the text classification accuracy on the GLUE benchmark (Wang et al., 2018).

BERT w/ LAs	SST2	WNLI	QNLI	MNLI	RTE	MRPC	QQP	Average
BERT (Baseline)	93.58	42.25	91.49	84.81	66.43	83.09	91.10	**78.96**
FLASH-Local	91.63	47.89	88.38	81.06	50.18	70.10	90.56	**74.26**
FLASH-Global	76.72	54.93	53.69	33.46	48.74	68.63	78.32	**59.21**
Linformer	81.54	56.34	63.06	67.54	45.13	68.38	81.32	**66.19**
Performer	80.16	45.07	60.77	39.81	45.49	67.40	75.88	**59.23**
TransNormer	81.88	56.34	67.67	67.01	53.07	70.10	83.13	**68.46**
YOSO	91.51	52.11	87.75	82.16	58.12	75.98	90.40	**76.86**
ReLU	81.77	56.34	61.54	70.14	47.29	69.85	82.44	**67.05**

Table 2. Evaluation of seven LAs on GPT-2 (Radford et al., 2019), a decoder-based LLM, with the text classification accuracy on the GLUE benchmark (Wang et al., 2018).

GPT-2 w/ LAs	SST2	WNLI	QNLI	MNLI	RTE	MRPC	QQP	Average
GPT-2 (Baseline)	91.28	57.75	88.39	81.54	60.65	74.51	89.13	**77.61**
FLASH-Local	83.60	53.52	77.16	73.97	48.01	68.87	86.40	**70.22**
FLASH-Global	50.92	50.70	54.27	34.59	52.35	68.38	63.19	**53.49**
Linformer	79.47	52.11	60.96	34.56	52.35	68.38	76.30	**60.59**
Performer	86.93	38.03	69.36	70.60	49.46	69.12	76.30	**65.69**
TransNormer	82.11	56.34	63.48	59.11	53.07	68.38	75.79	**65.47**
YOSO	88.42	45.07	82.23	77.80	54.51	73.04	87.72	**72.68**
ReLU	86.47	45.07	80.96	78.02	51.99	69.61	83.42	**70.79**

in Tabs. 1, 2, and 3, we have applied these LAs to their respective LLMs, assessing their performance across seven linguistic tasks from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018). To enhance comparison efficacy, we also report the accuracy of softmax-based LLMs as a baseline. This facilitates a straightforward evaluation of the average accuracy drop across the seven LAs and the seven tasks when being applied to different types of LLMs.

Result Analysis. Our evaluation shows that: (1) most LAs are effective in encoder-based LLMs, aligning with their initial design intent. However, their performance diminishes when applied to decoder-based or encoder-decoder-based LLMs. On average, seven LAs applied to encoder-based LLMs result in an average accuracy of 67.32, whereas for decoder-based or encoder-decoder-based models, the accuracy drops to 65.56 and 63.13, respectively; (2) as shown in Fig. 1 (left), advanced LA techniques perform well in encoder-based LLMs but struggle to replicate these results in decoder or encoder-decoder-based LLMs. For instance, FLASH-Local (Hua et al., 2022) and YOSO (Zeng et al., 2021) achieve score 74.26/76.86 on BERT, only slightly below the baseline, but drops to 70.22/72.68 on GPT-2 and further to 62.75/61.39 on T5, significantly lower than their softmax-based counterparts; (3) as shown in Fig. 1 (right), LAs that are less effective in encoder-based LLMs consistently underperform in decoder-based and encoder-decoder based LLMs, highlighting their distinct suitability for different LLM architectures.

Table 3. Evaluation of seven LAs on T5 (Raffel et al., 2020), an encoder-decoder-based LLM, with the text classification accuracy on the GLUE benchmark (Wang et al., 2018).

T5 w/ LAs	SST2	WNLI	QNLI	MNLI	RTE	MRPC	QQP	Average
T5 (Baseline)	93.81	36.62	91.73	86.54	58.12	80.64	90.89	**76.91**
FLASH-Local	77.87	56.34	58.87	49.44	52.71	68.38	75.62	**62.75**
FLASH-Global	80.62	56.34	63.65	49.87	46.93	68.38	79.29	**63.58**
Linformer	51.15	43.66	55.43	46.60	51.99	68.38	74.19	**55.91**
Performer	82.57	56.34	63.70	61.75	52.35	69.85	78.60	**66.45**
TransNormer	79.36	43.66	59.78	48.75	58.48	70.59	75.37	**62.29**
YOSO	78.33	56.34	59.55	48.64	47.65	68.38	70.87	**61.39**
ReLU	85.79	53.52	71.57	73.52	48.01	70.34	83.89	**69.52**

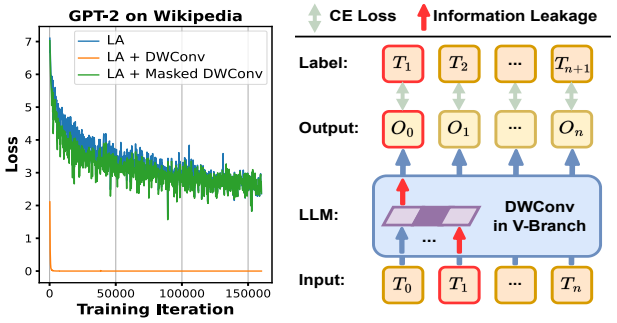

Figure 4. Existing augmented LAs fail in autoregressive LLMs. **Left**: The augmented DWConv branch results in zero loss/accuracy, as indicated by the yellow line. **Right**: Illustration of the information leakage phenomenon, i.e., next tokens are prematurely revealed as shown by red arrows, in autoregressive LLMs with DWConv in the $\mathbf{V}$ branch.

Limitations of Existing LAs. Our evaluation indicates that most LAs suffer an accuracy drop in autoregressive decoder-based LLMs in generation tasks. Advanced LA techniques, such as efficient depthwise convolution (DWConv) in the $\mathbf{V}$ (value) branch of attention modules (You et al., 2023b; Xiong et al., 2021), fail in autoregressive LLMs due to an information leakage from the inclusion of future context during training. As evident in Fig. 4, LA with DWConv convergences to zero loss early in training, but actual evaluation accuracy remains zero, indicating information leakage as also depicted in Fig. 4 (right). In addition, while LAs improve training and summarization, their effectiveness is limited in token-by-token generation and compatibility with speculative decoding to increase parallelism during generation remains challenging. We will further discuss our augmented methods for autoregressive LLMs and their integration with speculative decoding in subsequent sections.

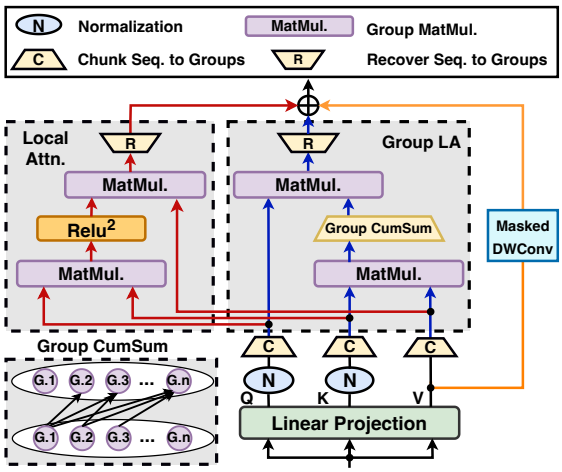

Figure 5. Model architecture of our LA augmentation.

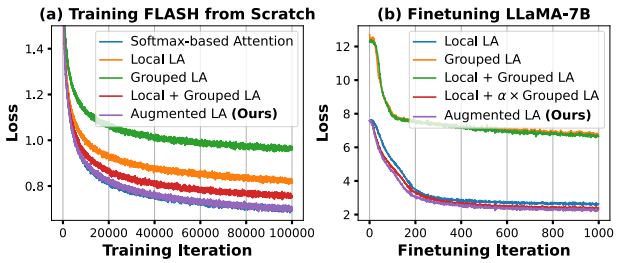

Figure 6. Tested our augmented linear attention mechanism for both training from scratch and fine-tuning from pre-trained model settings, where **(a)** shows the training progress of FLASH models (Hua et al., 2022); **(b)** depicts the fine-tuning performance of LLaMA-7B (Touvron et al., 2023a).

4. The Proposed Method

In this section, we introduce a revised local augmentation technique for existing LAs to enhance accuracy and examine the synergy of augmented LAs with speculative decoding for both efficient LLM training and autoregressive generation.

4.1. LA Augmentation for Autoregressive LLMs

Revised LA Augmentation. To address information leakage, we propose to design an effective masked DWConv instead of using a simple convolutional layer for enhancing the locality of the linear attention (You et al., 2023b; Xiong et al., 2021). Specifically, we adopt a causal mask on the DWConv layer to prevent tokens from accessing information from subsequent tokens, thereby preserving the inherent causality of the original attention mechanism, as illustrated in the right branch of Fig. 5. The masked DWConv prevents information leakage, leading to better loss convergence, as demonstrated in the left of Fig. 4. Unlike (Dauphin et al., 2017), our efficient DWConv is integrated directly into the attention block, not as a standalone component.

We build our DWConv augmentation on top of existing grouped LAs to speed up the linearized LLMs. The reason why we need the grouped LA is that standard LAs exhibit reduced efficiency in autoregressive settings due to the causal constraint (Hua et al., 2022). For example, the query vector $\mathbf{Q_t}$ at t-th time step interacts with the cumulative sum of all preceding results $\sum_{i=1}^{t} \mathbf{K_i V_i}$. This cumulative sum (`cumsum`) of $\mathbf{KV}$ product operations inherently creates a sequential dependency, and restricts the potential for parallel processing. To enhance efficiency, we partition the input sentence into non-overlapping groups. Within each group, we bypass local dependencies, allowing parallel processing. For interactions between groups, we only compute the cumulative sums at the group level for the $\mathbf{KV}$ products for improved efficiency, as depicted in the middle branch of Fig. 5. Furthermore, to improve local dependency handling, we employ parallel local attention within each group, using softmax-based attention, as depicted in the left branch of Fig. 5. The integration of this local attention strategy with our revised local augmentation contributes significantly to the performance, combining the efficiency of LAs with improved accuracy.

Verification on Small- and Large-Scale LLMs. We evaluate and verify the revised LA augmentation on both small- and large-scale LLMs, i.e., FLASH (Hua et al., 2022) and LLaMA-7B (Touvron et al., 2023a). For FLASH, we train a small model from scratch for 100K steps on enwik8 (Hutter, 2012). As shown in Fig. 6 (a), grouped LA leads to reduced accuracy or increased loss. Local LA alone is also ineffective. A combination of grouped and local LAs showed some improvement but remained inferior to the traditional softmax-based attention method. In contrast, our augmented LAs, blending the grouped LA concept with masked DWConv augmentation (with a kernel size of 63), achieved the most favorable results among all LAs, on par with the original softmax-based attentions. For LLama-7B, we finetune it using LAs on the RedPajama dataset (Computer, 2023) for 1K steps with a batch size of 64 following (Chen et al., 2023c). Fig. 6 (b) indicates a similar trend to FLASH, where local augmentation proves even more vital in this finetuning phase, and reliance solely on global LA leads to significantly higher loss. Note that we use a hyperparameter α to balance the interplay between global and local LAs. Overall, our augmented LAs combining the three branches in Fig. 5 consistently outperform existing LAs.

4.2. When LA Meets Speculative Decoding

To address limited parallellism in LLM serving, we aim to combine speculative decoding with our imporved LAs. However, direct integration is ineffective. Here, we explore

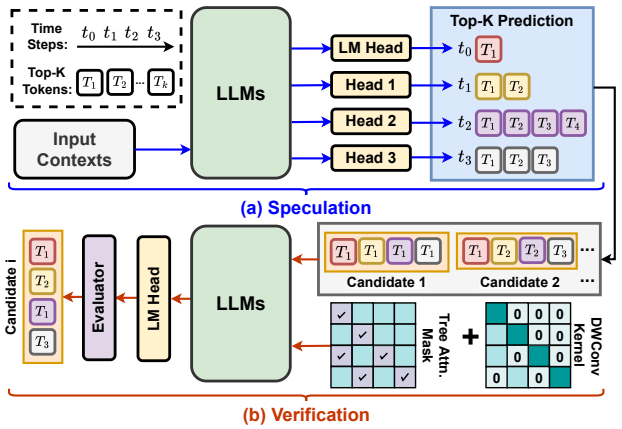

Figure 7. Illustrating the speculative decoding pipeline with our augmented LAs: (a) Speculation; and (b) Verification.

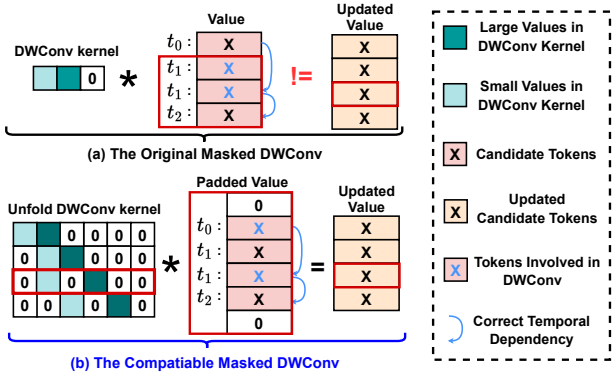

Figure 8. **(a)**: DWConv itself fails to capture the temporal dependency in speculative decoding; **(b)**: Our Unfolded DWConv kernels capture the correct temporal dependency.

compatibility challenges and propose seamless solutions.

Compatibility Analysis. Speculative decoding, such as Medusa (Cai et al., 2023b), uses smaller draft models to simultaneously predict multiple output tokens across different time steps, as illustrated in Fig. 7 (a). The original LLMs then act as verifiers, either accepting or rejecting them, and resampling if needed, as illustrated in Fig. 7 (b). This approach improves parallelism during LLM generation. However, combining LAs with speculative decoding is challenging because it generates multiple candidate outputs per step, with varying counts per time step, altering the temporal dependency. This change is not effectively captured by masked DWConvs and grouped LAs in our augmented LAs. As shown in Fig. 8 (a), using a masked DWConv with a kernel size of 3 to convolve over stacked candidate tokens at time step t_1 results in capturing time steps $\{t_1, t_1\}$, rather than the correct sequence $\{t_0, t_1\}$. This discrepancy occurs because, at time step t_1, two candidate tokens are included in the convolution instead of the final verified one, leading to a temporal misalignment.

Proposed Solution. To integrate our augmented LAs with the speculative decoding, we propose the updated design of DWConv and grouped LA to take into consideration the temporal dependencies represented in Medusa's tree-based attention mask. This design ensures the simultaneous processing of multiple candidate tokens while ensuring that each token only accesses information from its preceding token. As shown in Fig. 8 (b), we unfold the convolution into matrix multiplication, akin to the img2col method (Vasudevan et al., 2017). This unfolding allows for the integration of tree-based attention masks with DWConv kernels, addressing their compatibility with negligible overheads. For example, using a masked DWConv with an unfolded kernel to convolve over stacked candidate tokens at time step t_1 successfully captures the correct sequence $\{t_0, t_1\}$, while omitting an unchosen candidate at the same time step t_1. In addition, we categorize speculative tokens into groups based on temporal dependency, regardless of the number of candidates per time step. In this way, tokens in each group interact only with verified tokens from previous groups, aligning their visibility with the tree-based attention pattern.

5. Experiments

Models, Tasks, and Datasets. *Models.* We apply our proposed augmented LA on top of five models, including FLASH (Hua et al., 2022), T5 (Raffel et al., 2020), GPT-2 (Radford et al., 2019), LLaMA-2-7B, and LLaMA-2-13B (Touvron et al., 2023b). In particular, we train the FLASH (Hua et al., 2022) model of roughly 110M parameters from scratch and finetune the remaining language models of different sizes with our augmented LAs. *Tasks and Datasets.* For FLASH and LLaMA-2-7B/13B models, we evaluate them on language modeling tasks. Specifically, we train the FLASH model on the English partition of Wiki40b (Guo et al., 2020), which includes about 40B characters from 19.5M pages obtained from Wikipedia. We finetune the LLaMA-2-7B/13B models on RedPajama (Computer, 2023) dataset with about 1.2T tokens for 1K steps, following the setting of LongLora (Chen et al., 2023c). For T5 and GPT-2 models, we consider the text classification task to evaluate our augmented LAs, and choose seven datasets from GLUE (Wang et al., 2018) benchmark: SST2 (Socher et al., 2013), WNLI (Levesque et al., 2012), QNLI (Rajpurkar et al., 2016), MNLI (Williams et al., 2018), RTE (Dagan et al., 2006), MRPC (Dolan & Brockett, 2005), and QQP (DataCanary et al., 2017). In addition, we consider the evaluation of LLaMA models on six zero-shot or

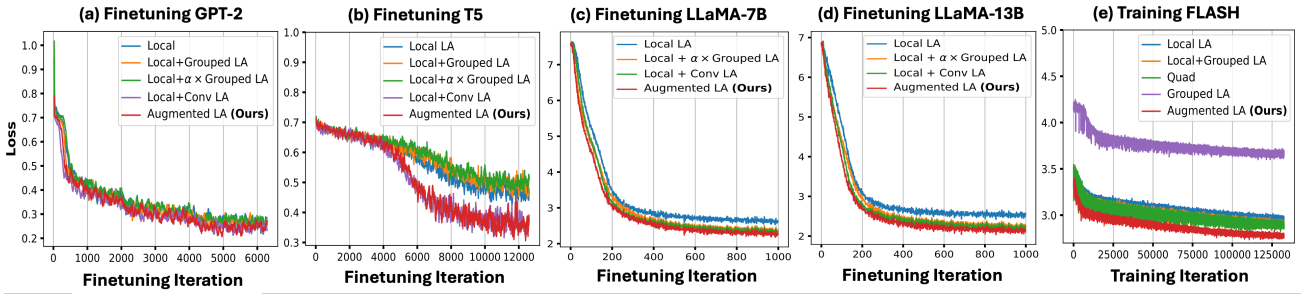

Figure 9. Visualizing the training trajectories of baseline LAs and our augmented LAs.

Table 4. Inference latency and memory comparison at various sequence lengths for LLaMA models.

Model	Attn.	Inference Latency (ms)					Inference Memory (GB)				
		2048	4096	8192	16384	32768	2048	4096	8192	16384	32768
LLaMA-2-7B	Original	302.3	812.6	2355.0	OOM	OOM	16.6	22.5	40.5	OOM	OOM
	Ours LA	275.2 (**-9%**)	529.8 (**-35%**)	1029.7 (**-56%**)	2032.9	3985.9	15.9 (**-4%**)	19.1 (**-15%**)	25.7 (**-37%**)	38.8	65.0
LLaMA-2-13B	Original	491.6	1319.7	3805.0	OOM	OOM	30.1	38.2	62.1	OOM	OOM
	Ours LA	449.4 (**-9%**)	876.5 (**-34%**)	1737.1 (**-54%**)	3460.9	OOM	29.1 (**-3%**)	33.8 (**-12%**)	43.2 (**-30%**)	61.9	OOM

few-shot downstream tasks: BBH (Suzgun et al., 2022), PIQA (Bisk et al., 2020), MMLU (Hendrycks et al., 2020), COPA (Wang et al., 2019), ARCC (Clark et al., 2018), and AGNews (Zhang et al., 2015). Following common evaluation settings, MMLU was tested with 5 shots, BBH with 3 shots, and the remaining tasks with zero shots.

Training Settings. *For the FLASH training task*, we train the model of roughly 110M parameters from scratch with a sequence length of 1024. The batch size is 256 and the token per batch is set to 2^{18}. We use the AdamW optimizer with linear learning rate decay and a peak learning rate of 7×10^{-4}, the momentum of the AdamW optimizer is set to $\beta_1 = 0.9$, $\beta_2 = 0.95$ and the group size is set to 256 following (Hua et al., 2022). *For the LLaMA-2 finetune task*, we train it for 1K steps with a peak learning rate of 2×10^{-5} and a batch size of 64. The learning rate scheduler is constant with 20 warmup steps. The optimizer is AdamW with the momentum of $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The group size is set to 64 following (Chen et al., 2023c). *For the GLUE task*, We finetune the models for 3 epochs with a learning rate of 2×10^{-5} and a batch size of 32 (Devlin et al., 2018). The group size is set to 64, and the sequence length is set to 256.

Baseline and Evaluation Metrics. *Baselines*. For the text classification task on the GLUE benchmark, we compare the proposed augmented LAs with FLASH-Local&Global (Hua et al., 2022), Linformer (Wang et al., 2020), Performer (Choromanski et al., 2021), TransNormer (Qin et al., 2022), YOSO (Zeng et al., 2021), ReLU (Cai et al., 2023a). For the LLaMA-2 finetune tasks, we compare the proposed augmented LAs with the local and global attention proposed in (Hua et al., 2022), i.e., FLASH-Local/Global. For the FLASH training task, we compare our proposed method with local, global, and quadratic softmax-based attention. *Evaluation Metrics*. For the GLUE task, we use the classification accuracy to evaluate the augmented LA and baselines. For the LLaMA-2 finetune task, we use the perplexity on PG-19 (Rae et al., 2019) to evaluate all methods. For the FLASH training task, we use the validation set perplexity of Wiki40B to evaluate. In addition, to evaluate the speedups after integrating our LAs and speculative decoding, we test the decoding speeds on MT-Bench (Zheng et al., 2023) following (Cai et al., 2023b).

5.1. Our Linearized LLMs over Original LLMs

We analyze the latency improvements and memory efficiencies of our linearized LLMs compared to conventional LLMs. As detailed Tab. 4, our approach reduces latency by up to 56.3% and memory usage by 36.5% for models like LLaMA-2-7B and LLaMA-2-13B on A100-80G device. In addition, our linearized LLMs extend the maximum supported sequence lengths from 8K to 32K for LLaMA-2-7B on the same GPU, demonstrating our method's efficacy and scalability in large-scale models. We also provide detailed reports on latency and memory consumption for the LLaMA-2-7B model across four downstream tasks, under varied prefill and decode size configurations (see Appendix C for details). As shown in Tab. 5, our approach reduces latency by up to 39.1% and memory usage by up to 32.8% during runtime when deploying LLaMA-2-7B models on a A100-80G GPU. In addition, we compare the

Table 5. Inference latency and memory comparison at various task prefill and decode sizes for LLaMA-2-7B models.

	Attn.	Prefill and Decode Sequence Lengths			
		(340, 160)	(60, 20)	(7000, 200)	(1700, 400)
Latency (ms)	Original	325.00	40.61	709.59	894.21
	Ours LA	**290.08**	**37.51**	**432.48**	**736.51**
Memory (GB)	Original	13.4	12.8	32.3	15.7
	Ours LA	**13.1**	**12.8**	**21.7**	**14.8**

Table 6. Accuracy comparison on six zero/few-shot downstream tasks under 0.8s latency (sequence length is 4K).

LLaMA-2	Attn.	BBH	PIQA	MMLU	COPA	ARCC	AGNews
7B	Original	33.50	63.22	**45.40**	85.00	**52.17**	78.17
13B	**Ours LA**	**33.91**	**68.06**	36.57	**85.00**	51.74	**78.95**

Table 7. Accuracy comparison using the LLaMA-2-7B model on six zero-shot or few-shot downstream tasks.

Attn.	BBH	PIQA	MMLU	COPA	ARCC	AGNews	Lat.
FLASH-Local	30.89	61.65	34.21	68.60	45.01	69.05	0.5s
FLASH-Global	31.78	61.48	35.62	75.00	47.36	78.14	0.5s
Aug. FLASH	**32.70**	**62.52**	**36.04**	**78.00**	**48.36**	**78.20**	0.5s

Table 8. Throughput of LLaMA (tokens/s) with LAs and the speculative decoding on MT-Bench (Zheng et al., 2023).

LLaMA w/	Loc.	Loc.+Gro.	Loc.+Conv	Loc.+Gro.+Conv
7B	32.3 (**1.0x**)	26.8 (**1.0x**)	30.4 (**1.0x**)	25.9 (**1.0x**)
7B w/ Spec.	63.3 (**2.0x**)	50.5 (**1.9x**)	55.1 (**1.8x**)	50.7 (**2.0x**)
13B	26.1 (**1.0x**)	22.7 (**1.0x**)	22.3 (**1.0x**)	20.4 (**1.0x**)
13B w/Spec.	54.4 (**2.1x**)	42.6 (**1.9x**)	47.0 (**2.1x**)	41.7 (**2.0x**)

accuracy of our augmented linear attention method and the original attention-based LLaMA models under comparable inference latency on six downstream tasks. As shown in Tab. 6, LLaMA-2-13B with our augmented linear attention, achieves comparable inference latency to the original LLaMA-2-7B at a 4K sequence length, while outperforming the original LLaMA-2-7B in four out of six downstream tasks. These results validate that our method can boost downstream task performance.

5.2. Our Augmented LAs over SOTA LA Baselines

Overall Comparison. We apply our augmented LAs to five decoder-based or encoder-decoder-based LLMs and compare them with other LA baselines. The training trajectories are visualized in Fig. 9. We see that our augmented LAs consistently achieve a better convergence loss as compared to all baselines. As for the quantitative results:

1. *Text Classification with GPT-2 and T5.* We evaluate the performance of GPT-2 and T5 with our augmented LAs on the GLUE benchmark, with results provided in Appendix F. Our augmented LAs consistently yield better accuracy, achieving an average increase of 1.87 percentage points in classification accuracy on the GLUE benchmark compared to competitive existing LA baselines, such as FLASH-Local and FLASH-Global.

2. *Language Modeling with FLASH and LLaMA-7B/13B.* We evaluate the perplexity of LLaMA-7B/13B with our augmented LAs on PG-19, with results provided in Appendix G. The results on LLaMA models reveal that our augmented LAs with both the local augmentation and grouped LAs outperform all baselines, resulting in a 6.67/6.33 reduction in perplexity. The results on FLASH models consistently validate the effectiveness of our augmented LAs, leading to 1.49 to 20.09 perplexity reductions as compared to other LAs and even 0.24 reduction over original attention. The effectiveness of our augmented LAs is consistently validated by results on FLASH models and the Wiki40B dataset, demonstrating perplexity reductions ranging from 1.49 to 20.09 as compared to baselines, and even a 0.24 reduction over the original attention.

3. *Downstream Tasks on LLaMA-2-7B.* We analysis six downstream tasks: BBH, PIQA, MMLU, COPA, ARCC, and AGNews. Using standard evaluation settings, MMLU was tested with 5 shots, BBH with 3 shots, and the remaining tasks with zero shots. As shown in Tab. 7, our augmented linear attention not only reduces perplexity but also improves accuracy across all tasks. Specifically, with models like FLASH, our method achieved an average accuracy improvement of 3.53%.

In addition, we extend our methods to three more linear attention methods, with summarized results in Appendix H.

Generation Speedups by Integrating LAs with Speculative Decoding. We benchmark the speedups of our compatible LAs with speculative decoding. As shown in Tab. 8, we test the LLaMA-7B/13B models which are adapted into a chat model format, similar to LongLora (Chen et al., 2023c). Following Medusa (Cai et al., 2023b), we train Medusa heads for speculative decoding. Speed tests for the 7B and 13B models are conducted on a single A100-80GB GPU, we observe that our revised LAs are compatible with speculative decoding and approximately doubled the speed.

Table 9. Comparison of our method with the integration of FLASH (Hua et al., 2022) and Medusa (Cai et al., 2023b).

Methods	Total Latency	Attention	FFNs	Others
FLASH + Medusa	137.2 ms	119.7 ms	8.2 ms	9.3 ms
Ours Aug. LA	49.7 ms **(-64%)**	32.2 ms	8.2 ms	9.3 ms

5.3. Ablation Study

Comparison with Direct Integration. To verify the effectiveness of our causal and compatible augmentation techniques, we compare them with the direct integration of previous linear attention FLASH (Hua et al., 2022) and the speculative decoding method Medusa (Cai et al., 2023b). As shown in Tab. 9, our method applied to LLaMA-2-7B models on A100 GPUs for a single batch of speculative decoding (64 speculated tokens and 42 sequence candidates), achieves a 64% reduction in total latency compared to the direct integration, while also reducing QKV memory requirements by 75% from 0.4 GB to 0.1 GB.

Our techniques outperform direct integration because standard implementations, even with linear attention like FLASH and speculative decoding like Medusa, face two key limitations without our augmentations: (1) slow sequence-based decoding and (2) lack of optimizations such as shared cumulative sum (`cumsum`) and key-value (KV) states for batch processing. Conventional strategies for compatible KV caching rely on sequence-based decoding, assigning distinct KV caches to each speculated sequence candidate, as shown in Fig. 7. This results in unnecessary computational effort and memory inefficiency since candidates with identical prefixes are processed separately. In contrast, our method addresses these issues by ensuring identical prefixes are computed only once, mitigating these issues with time-dependent causal and compatible augmentation in linear attention and speculative decoding.

Our LA Speedups. We benchmarked the training speed of FLASH using both the original attention and our augmented LAs, with a batch size of 1, on a single A100-40G GPU. Our results show that the augmented LAs significantly improve training speed. For sequence lengths of 4K and 8K, they are 1.52× and 2.94× faster, respectively. FLASH with augmented LAs takes 1.05 seconds and 1.95 seconds per training step for 4K and 8K sequences, compared to 1.60 seconds and 5.74 seconds with the original attention. The group size in FLASH was consistently set to 256.

Extend to Longer Sequence. We fine-tuned LLaMA-2-7B to extend its sequence length from 4K to 8K using our augmented LAs, following LongLora (Chen et al., 2023c) setting on the RedPajama dataset. For a fair comparison, we used only the local attention in LongLora, maintaining a block size of 256. Our augmented LAs reduced perplexity from 15.29 to 13.86, demonstrating their effectiveness in handling longer sequences.

6. Conclusion

This paper presents the first empirical analysis of linearized autoregressive LLMs, revealing significant limitations of existing linear attention methods in effectively handling masked attention and integration with speculative decoding. To address these challenges, we introduced an approach that combines effective local augmentation with seamless compatibility for speculative decoding. Our experiments across a range of LLMs consistently demonstrate that our method achieves substantial performance gains. Notably, we achieve up to a 6.67 perplexity reduction and up to 2× speedups in generation compared to existing linear attention methods. Our work paves the way for more efficient training and deployment of powerful autoregressive LLMs, especially for long-sequence applications.

Acknowledgements

This work is supported by the National Science Foundation (NSF) EPCN program (Award number: 1934767) and the CoCoSys, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. We extend our gratitude towards Arthur Szlam, Marc'aurelio Ranzato, and Cliff Young for reviewing the paper and providing insightful feedback. We also thank the extended team at Google DeepMind, who enabled and supported this research direction.

Impact Statement

Efficient LLM Training and Serving Goal. The recent advancements in Large Language Models (LLMs), exemplified by OpenAI's GPT-3 with its 175 billion parameters, have underscored the significant data and computational power required for such technologies. Training models of this scale incur substantial costs, both financially and environmentally. For instance, the cost necessary to train GPT-3 could exceed 4 million equivalent GPU hours (Brown et al., 2020), and the carbon footprint of training a single Transformer model might rival the lifetime emissions of five average American cars (Strubell et al., 2019). Addressing the challenges of efficient training and serving of LLMs is therefore not only a technical imperative but also an environmental and ethical necessity.

Societal Consequences. The success of this project in enabling more efficient training and serving of LLMs will have far-reaching implications, especially in processing long sequences commonly encountered in document handling. Our

efforts are set to substantially influence various societal and economic sectors. The enhanced efficiency of LLMs promises transformative changes in diverse applications ranging from document summarization and question answering to personal digital assistants, security, and augmented reality. The development and exploration of linearized LLMs mark a pivotal progress in rendering these models both more accessible and environmentally sustainable.

References

Agrawal, A., Kedia, N., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B. S., Tumanov, A., and Ramjee, R. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve. arXiv preprint arXiv:2403.02310, 2024.

Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: A Family of Highly Capable Multimodal Models. arXiv preprint arXiv:2312.11805, 2023.

Arar, M., Shamir, A., and Bermano, A. H. Learned Queries for Efficient Local Attention. In CVPR, 2022.

Bae, S., Ko, J., Song, H., and Yun, S.-Y. Fast and Robust Early-Exiting Framework for Autoregressive Language Models with Synchronized Parallel Decoding. arXiv preprint arXiv:2310.05424, 2023.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. PIQA: Reasoning about Physical Commonsense in Natural Language. In AAAI, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language Models are Few-shot Learners. NeurIPS, 2020.

Cai, H., Li, J., Hu, M., Gan, C., and Han, S. EfficientViT: Lightweight Multi-Scale Attention for High-Resolution Dense Prediction. In ICCV, 2023a.

Cai, T., Li, Y., Geng, Z., Peng, H., and Dao, T. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. arXiv preprint arXiv:2401.10774, 2023b.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating Large Language Model Decoding with Speculative Sampling. arXiv preprint arXiv:2302.01318, 2023a.

Chen, J., Zhu, D., Shen, X., Li, X., Liu, Z., Zhang, P., Krishnamoorthi, R., Chandra, V., Xiong, Y., and Elhoseiny, M. MiniGPT-v2: Large Language Model as a Unified Interface for Vision-Language Multi-Task Learning. arXiv preprint arXiv:2310.09478, 2023b.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. arXiv preprint arXiv:2309.12307, 2023c.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking Attention with Performers. In ICLR, 2021.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think You Have Solved Question Answering? Try Arc, the AI2 Reasoning Challenge. arXiv preprint arXiv:1803.05457, 2018.

Computer, T. RedPajama: An Open Dataset for Training Large Language Models, October 2023. URL https://github.com/togethercomputer/RedPajama-Data. Software.

Dagan, I., Glickman, O., and Magnini, B. The PASCAL Recognising Textual Entailment Challenge. In Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment. Springer Berlin Heidelberg, 2006.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and Memory-efficient Exact Attention with IO-Awareness. NeurIPS, 2022.

DataCanary, hilfialkaff, Jiang, L., Risdal, M., Dandekar, N., and tomtung. Quora Question Pairs, 2017. URL https://kaggle.com/competitions/quora-question-pairs.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language Modeling with Gated Convolutional Networks. In ICML, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805, 2018.

Dolan, W. B. and Brockett, C. Automatically Constructing a Corpus of Sentential Paraphrases. In IWP, 2005. URL https://aclanthology.org/I05-5002.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In ICLR, 2021.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. OPTQ: Accurate Quantization for Generative Pre-trained Transformers. In The Eleventh ICLR, 2022.

Guo, M., Dai, Z., Vrandečić, D., and Al-Rfou, R. Wiki-40B: Multilingual Language Model Dataset. In LREC, 2020.

Harma, S. B., Chakraborty, A., Kostenok, E., Mishin, D., Ha, D., Falsafi, B., Jaggi, M., Liu, M., Oh, Y., Subramanian, S., and Yazdanbakhsh, A. Effective Interplay between Sparsity and Quantization: From Theory to Practice. arXiv preprint arXiv:2405.20935, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring Massive Multitask Language Understanding. arXiv preprint arXiv:2009.03300, 2020.

Hua, W., Dai, Z., Liu, H., and Le, Q. Transformer Quality in Linear Time. In ICML, 2022.

Hutter, M. The Human Knowledge Compression Contest. URL http://prize. hutter1. net, 2012.

Kao, S.-C., Subramanian, S., Agrawal, G., Yazdanbakhsh, A., and Krishna, T. FLAT: An Optimized Dataflow for Mitigating Attention Bottlenecks. In ASPLOS, 2023.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In ICML, 2020.

Kim, S., Mangalam, K., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Big Little Transformer Decoder. arXiv preprint arXiv:2302.07863, 2023.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient Memory Management for Large Language Model Serving with PagedAttention. In SOSP, 2023.

Lee, H., Kim, J., Willette, J., and Hwang, S. J. SEA: Sparse Linear Attention with Estimated Attention Mask. arXiv preprint arXiv:2310.01777, 2023.

Levesque, H., Davis, E., and Morgenstern, L. The winograd Schema Challenge. In KR, 2012.

Leviathan, Y., Kalman, M., and Matias, Y. Fast Inference from Transformers via Speculative Decoding. In ICML, 2023.

Lin, C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out, 2004.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In ICCV, 2021.

Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., and Zhang, L. SOFT: Softmax-free Transformer with Linear Complexity. NeurIPS, 2021.

Ma, X., Fang, G., and Wang, X. LLM-Pruner: On the Structural Pruning of Large Language Models. arXiv preprint arXiv:2305.11627, 2023.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. Building a Large Annotated Corpus of English: The Penn Treebank. Computational linguistics, 1993.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer Sentinel Mixture Models. In ICLR, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Wong, R. Y. Y., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. SpecInfer: Accelerating Generative LLM Serving with Speculative Inference and Token Tree Verification. arXiv preprint arXiv:2305.09781, 2023.

OpenAI. ChatGPT: Language Model for Dialogue Generation, 2023a. URL https://www.openai.com/chatgpt/.

OpenAI. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774, 2023b.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. Random Feature Attention. arXiv preprint arXiv:2103.02143, 2021.

Qin, Z., Han, X., Sun, W., Li, D., Kong, L., Barnes, N., and Zhong, Y. The Devil in Linear Transformer. arXiv preprint arXiv:2210.10340, 2022.

Qin, Z., Li, D., Sun, W., Sun, W., Shen, X., Han, X., Wei, Y., Lv, B., Yuan, F., Luo, X., et al. Scaling Transnormer to 175 Billion Parameters. arXiv preprint arXiv:2307.14995, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language Models are Unsupervised Multitask Learners. OpenAI blog, 2019.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive Transformers for Long-range Sequence Modelling. arXiv preprint arXiv:1911.05507, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. JMLR, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In EMNLP, 2016.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot Text-to-Image Generation. In ICML, 2021.

Roberts, A., Chung, H. W., Levskaya, A., Mishra, G., Bradbury, J., Andor, D., Narang, S., Lester, B., Gaffney, C., Mohiuddin, A., Hawthorne, C., Lewkowycz, A., Salcianu, A., van Zee, M., Austin, J., Goodman, S., Soares, L. B., Hu, H., Tsvyashchenko, S., Chowdhery, A., Bastings, J., Bulian, J., Garcia, X., Ni, J., Chen, A., Kenealy, K., Clark, J. H., Lee, S., Garrette, D., Lee-Thorp, J., Raffel, C., Shazeer, N., Ritter, M., Bosma, M., Passos, A., Maitin-Shepard, J., Fiedel, N., Omernick, M., Saeta, B., Sepassi, R., Spiridonov, A., Newlan, J., and Gesmundo, A. Scaling Up Models and Data with `t5x` and `seqio`. arXiv preprint arXiv:2203.17189, 2022. URL https://arxiv.org/abs/2203.17189.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V., Tay, Y., and Metzler, D. Confident Adaptive Language Modeling. NeurIPS, 2022.

See, A., Liu, P. J., and Manning, C. D. Get To The Point: Summarization with Pointer-Generator Networks. In ACL, 2017.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In EMNLP, 2013.

Strubell, E., Ganesh, A., and McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. arXiv preprint arXiv:1906.02243, 2019.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., et al. Challenging Big-Bench Tasks and Whether Chain-of-Thought can Solve them. arXiv preprint arXiv:2210.09261, 2022.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford Alpaca: An Instruction-Following LLaMA Model, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. LLaMA: Open and Efficient Foundation Language Models. arXiv preprint arXiv:2302.13971, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. LLaMA 2: Open Foundation and Fine-tuned Chat Models. arXiv preprint arXiv:2307.09288, 2023b.

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. Maxvit: Multi-axis Vision Transformer. In ECCV, 2022.

Vasudevan, A., Anderson, A., and Gregg, D. Parallel Multi Channel Convolution using General Matrix Multiplication. In ASAP, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is All you Need. In NeurIPS, 2017.

Waisberg, E., Ong, J., Masalkhi, M., Zaman, N., Sarker, P., Lee, A. G., and Tavakkoli, A. Google's AI chatbot "Bard": A Side-by-Side Comparison with ChatGPT and its Utilization in Ophthalmology. Eye, 2023.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A Multi-task Benchmark and Analysis Platform for Natural Language Understanding. arXiv preprint arXiv:1804.07461, 2018.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. SuperGLUE: A Stickier Benchmark for General-purpose Language Understanding Systems. NeurIPS, 2019.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-Attention with Linear Complexity. arXiv preprint arXiv:2006.04768, 2020.

Williams, A., Nangia, N., and Bowman, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In NAACL, 2018.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant: Accurate and Efficient Post-training Quantization for Large Language Models. In ICML, 2023.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A Nyström-based Algorithm for Approximating Self-attention. In AAAI, 2021.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated Linear Attention Transformers with Hardware-efficient Training. arXiv preprint arXiv:2312.06635, 2023.

You, H., Sun, Z., Shi, H., Yu, Z., Zhao, Y., Zhang, Y., Li, C., Li, B., and Lin, Y. ViTCoD: Vision Transformer Acceleration via Dedicated Algorithm and Accelerator Co-Design. In 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 273–286. IEEE, 2023a.

You, H., Xiong, Y., Dai, X., Wu, B., Zhang, P., Fan, H., Vajda, P., and Lin, Y. C. Castling-ViT: Compressing Self-Attention via Switching Towards Linear-Angular Attention at Vision Transformer Inference. In CVPR, 2023b.

You, H., Shi, H., Guo, Y., and Lin, Y. ShiftAddViT: Mixture of Multiplication Primitives Towards Efficient Vision Transformer. Advances in Neural Information Processing Systems, 36, 2024.

Zeng, Z., Xiong, Y., Ravi, S., Acharya, S., Fung, G. M., and Singh, V. You Only Sample (almost) Once: Linear Cost Self-attention via Bernoulli Sampling. In ICML, 2021.

Zhang, X., Zhao, J., and LeCun, Y. Character-level Convolutional Networks for Text Classification. NeurIPS, 2015.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv preprint arXiv:2306.05685, 2023.

Zhu, C., Ping, W., Xiao, C., Shoeybi, M., Goldstein, T., Anandkumar, A., and Catanzaro, B. Long-short Transformer: Efficient Transformers for Language and Vision. NeurIPS, 2021.

Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. arXiv preprint arXiv:2304.10592, 2023.

A. Comprehensive Related Works

Autoregressive LLMs. Transformers (Vaswani et al., 2017; Dosovitskiy et al., 2021) have significantly advanced the fields of language and vision, leading to the development of foundation LLMs such as ChatGPT (Brown et al., 2020; OpenAI, 2023b), LLaMA (Touvron et al., 2023a;b), Gemini (Anil et al., 2023), DALL-E (Ramesh et al., 2021), etc. To date, various Transformers have emerged to serve distinct needs, broadly categorized into three types: *encoder-based*, *decoder-based*, and *encoder-decoder* models. Encoder-based models like BERT (Devlin et al., 2018) focus on natural language understanding and are also commonly used in image processing (Dosovitskiy et al., 2021). Encoder-decoder models like the original Transformer (Vaswani et al., 2017), Bard (Waisberg et al., 2023), and T5 (Raffel et al., 2020; Roberts et al., 2022) are designed for sequence-to-sequence tasks (e.g., translation, speech recognition), where the encoder extracts features and the decoder produces outputs based on these features. Decoder-based models, including GPT (Radford et al., 2019; OpenAI, 2023b) and LLaMA (Touvron et al., 2023a), generate text sequentially by predicting the next token based on previous ones. All these models leverage Transformer architectures but differ in their specific purposes and structures. Both encoders and decoders are leveraged in multimodal models like MiniGPT (Zhu et al., 2023; Chen et al., 2023b) and DALL-E (Ramesh et al., 2021). Note that the model architectures used in all categories are based on Transformer. The primary difference lies in their purpose: the encoder is designed to extract features, while the decoder focuses on scoring and generating outputs. Our work presents a comprehensive study of applying linear attention techniques to the encoder/decoder-based LLMs.

Efficient Linear Attention. Transformers' self-attention modules, known for their quadratic computational complexity (Zhu et al., 2021; Katharopoulos et al., 2020), have spurred the development of linear attention methods to improve efficiency, especially in encoder-based LLMs for better training and inference. Techniques such as local attentions (Liu et al., 2021; Arar et al., 2022; Wang et al., 2020; Tu et al., 2022; You et al., 2023a) limit self-attention to neighboring tokens or group attention queries to reduce the computational cost, while kernel-based linear attentions (Liu et al., 2021; Arar et al., 2022; Wang et al., 2020; Tu et al., 2022; You et al., 2024) decompose the softmax with kernel functions and exchange the computation order. However, only a few linear attention approaches focus on decoder-based autoregressive LLMs, aiming to reduce RNN-style sequential state updates over a large number of steps (Hua et al., 2022; Katharopoulos et al., 2020). Recent studies, like LongLoRA (Chen et al., 2023c), aim to adapt local attention techniques for efficient fine-tuning of pre-trained autoregressive LLMs, yet a thorough analysis comparing various linear attention methods for autoregressive LLMs remains lacking. This paper uniquely provides a systematic review of existing linear attentions for decoder-based autoregressive LLMs and investigates how to efficiently enhance less effective linear attention methods.

Speculative Decoding. Linear attention techniques alleviate the training inefficiency in LLMs by mitigating the quadratic complexity with regard to the number of input tokens. However, during deployment, autoregressive decoding necessitates sequential token-by-token text generation, which curtails parallelism and restricts the number of input tokens. Speculative decoding (Chen et al., 2023a; Miao et al., 2023; Kim et al., 2023; Leviathan et al., 2023; Cai et al., 2023b) has proven to be an effective strategy for boosting parallelism in LLM serving, utilizing small speculative models for initial generation, with original LLMs serving as validators to assess if the output meets standards or needs resampling. Recent works like Medusa (Cai et al., 2023b) further argue that the small speculative models and LLMs can be the same model, and other studies (Schuster et al., 2022; Bae et al., 2023) suggest using shallow layers for generation and deeper layers for verification, based on early exit strategies. Such speculative decoding and linear attention jointly ensure efficient LLM training and generation, especially for long sequence inputs. In this paper, we take the initiative to investigate the synergy between linearized LLMs and speculative sampling, to improve the efficiency of training and serving LLMs.

B. More Visualization of Training Trajectories.

As detailed in Sec. 5.3, we present a quantitative analysis comparing local LAs, grouped LAs, and our augmented LAs that combine both local augmentation and grouped LAs. This appendix provides the training trajectories for GPT-2 using these LA methods. Fig. 10 demonstrates that our local augmentation, specifically masked DWConv, effectively enhances both local and grouped LAs. Moreover, our augmented LAs, which integrate local augmentation with grouped LAs, exhibit the most favorable convergence in terms of loss.

C. More Profiling on the LLaMA-2-7B Model

We provide detailed profiling and comparisons below to illustrate the runtime distribution between attention and feed-forward networks (FFNs), highlighting that attention is a bottleneck even for LLMs with 7B parameters. To ensure a real-world

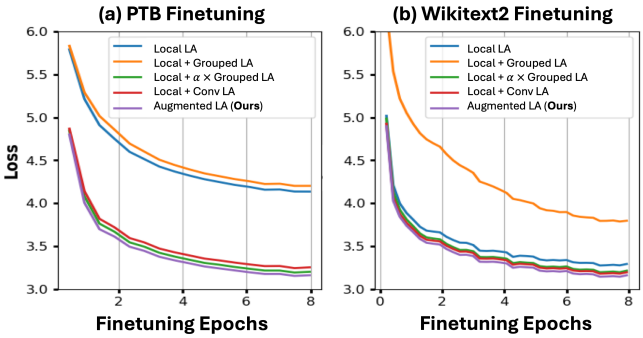

Figure 10. Visualizing the training trajectories of baseline LAs and our augmented LAs.

application scenario, we profiled the LLaMA-2-7B model across four settings of prefill and decode sizes, adhering to benchmarks commonly used in academia and industry, as summarized in Tab. 10.

Table 10. Dataset and task details for different prefill and decode size settings.

(Prefill, Decode)	Task	Dataset	Referenced Paper
(340, 160)	Chat	ShareGPT	(Kwon et al., 2023)
(60, 20)	Chat	Stanford Alpaca	(Taori et al., 2023)
(7000, 200)	Summarization	ArXiv Summarization	(Agrawal et al., 2024)
(1700, 400)	Chat	OpenChat ShareGPT 4	(Agrawal et al., 2024)

As shown in Tab. 11, profiling the LLaMA-2-7B models under the four prefill and decode size settings reveals that the average runtime latency attributed to attention and FFNs accounts for 55% and 21% of the total runtime across these settings, respectively. This indicates that although FFNs are a bottleneck in the model, attention is an even more significant bottleneck, especially for large-scale LLMs and extended dialogue sequences (e.g., 67.8% runtime latency for the arxiv summarization task). Therefore, optimizing attention blocks can yield considerable speed improvements, particularly for tasks with large prefill or decode sequence lengths. This is corroborated by contemporary studies on linear attention-based LLMs (Lee et al., 2023; Yang et al., 2023) and efforts to optimize attention, such as FlashAttention (Dao et al., 2022) and FLAT (Kao et al., 2023).

Table 11. Latency breakdown of LLaMA-2-7B models under different prefill and decode size settings.

(Prefill, Decode)	(340, 160)	(60, 20)	(7000, 200)	(1700, 400)
Attention (ms)	158.97 (**48.9%**)	20.12 (**49.5%**)	481.35 (**67.8%**)	481.41 (**53.8%**)
FFNs (ms)	74.64 (**23.0%**)	9.22 (**22.7%**)	111.90 (**15.8%**)	188.98 (**21.1%**)
Others (ms)	91.39 (**28.1%**)	11.27 (**27.8%**)	116.34 (**16.4%**)	223.83 (**25.1%**)
Total Latency (ms)	325.00	40.61	709.59	894.21

D. Breakdown Analysis of Our Augmented LAs

To gain insights into the contribution of each component in our augmented LAs, we show the breakdown analysis using GPT-2 and T5 models on Wikitext2 (Merity et al., 2017)/PTB (Marcus et al., 1993) and CNN/Daily Mail (See et al., 2017) datasets, respectively. As shown in Tabs. 12 and 13, our local augmentation, i.e., masked DWConv, consistently augments the local or grouped LAs, leading to 5.71 perplexity reductions on GPT-2 and 3.59 Rouge1 score (Lin, 2004) improvement on T5. Our augmented LAs, consisting of both local augmentation and grouped LAs, achieve the best results, i.e., 11.83∼17.54 perplexity reduction and 4.23∼15.45 Rouge1 score improvement, over all other LA variants.

Table 12. Perplexity of GPT-2 with our augmented LAs on the Wikitext2 and PTB datasets.

GPT-2 w/	Loc.	Loc.+Gro.	Loc.**+Conv**	**Augmented LA**
Wikitext2	56.80	42.81	51.09	39.26
PTB	69.32	57.72	84.24	46.85

Table 13. Ablation studies of fine-tuning T5 with LAs on the CNN/Daily Mail dataset (See et al., 2017).

T5 w/	**Rouge1**	**Rouge2**	**RougeL**	**RougeLsum**
Local LA	8.65	0.17	7.14	8.27
Grouped LA	6.14	0.86	5.77	5.50
Local + Grouped LA	19.87	3.07	14.54	18.29
Local + $\alpha\times$Grouped LA	19.01	2.90	13.99	17.54
Local LA + DWConv	12.24	0.20	8.95	11.38
Augmented LAs	24.10	4.93	17.22	22.11

E. Additional Training and Evaluation Settings and Model Hyperparameters.

Model, Task, Dataset. *Model:* We evaluate seven existing linear attention methods on top of three representative Transformers: (1) encoder-based BERT model of 12 layers and around 400M parameters, (2) decoder-based GPT-2 model of 12 layers and around 500M parameters, and (3) encoder-decoder-based T5 model of 12 layers and around 900M parameters. *Task and Dataset:* We conduct the evaluation on the text classification task across seven linguistic tasks from the General Language Understanding Evaluation (GLUE) benchmark: SST2, WNLI, QNLI, MNLI, RTE, MRPC, and QQP.

Training and Evaluation Settings. We fine-tuned all models for 3 epochs with a sequence length of 256, using a learning rate of 2×10^{-5} and a batch size of 32. The optimizer was AdamW, with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ following the standard training recipe in (Devlin et al., 2018). For the GLUE task, classification accuracy was utilized to evaluate the performance of all linear attention methods.

F. Text Classification with GPT-2 and T5

We evaluate the performance of GPT-2 and T5 with our augmented LAs on the GLUE benchmark. As shown in Tab. 14, our augmented LAs consistently yield better accuracy, achieving an average increase of 1.87 percentage points in classification accuracy on the GLUE benchmark compared to competitive existing LA baselines, such as FLASH-Local/Global.

Table 14. Evaluation of augmented LAs on T5 and GPT-2, with the classification accuracy on the GLUE benchmark.

GPT-2 w/	**SST2**	**WNLI**	**QNLI**	**MNLI**	**RTE**	**MRPC**	**QQP**	**Average**
LA Baseline	83.60	53.52	77.16	73.97	48.01	68.87	86.40	**70.22**
Loc.+Gro.	82.34	46.48	79.11	75.09	50.20	68.38	86.16	69.68
Loc.+α*Gro.	83.72	54.04	79.15	73.76	46.68	69.61	86.11	70.44
Augmented LA	84.72	54.93	80.01	74.26	50.90	69.85	86.16	**71.55**
T5 w/	**SST2**	**WNLI**	**QNLI**	**MNLI**	**RTE**	**MRPC**	**QQP**	**Average**
LA Baseline	77.87	56.34	58.87	49.44	52.71	68.38	75.62	**62.75**
Loc.+Gro.	76.95	56.34	60.37	51.20	49.82	68.38	75.44	62.64
Loc.+α*Gro.	78.10	56.34	59.62	51.49	49.10	68.38	75.62	62.66
Augmented LA	82.00	56.34	59.78	54.26	54.15	68.38	76.68	**64.51**

G. Language Modeling with FLASH and LLaMA-7B/13B

We evaluate the perplexity of LLaMA-7B/13B with our augmented LAs on PG-19. As shown in Tab. 15, integrating our local augmentation, i.e., masked DWConv, with the local LAs results in a 6.67/6.33 reduction in perplexity. The results on LLaMA models reveal that our augmented LAs with both the local augmentation and grouped LAs outperform all baselines, resulting in a 6.67/6.33 reduction in perplexity. The results on FLASH models and the Wiki40B dataset consistently validate the effectiveness of our augmented LAs, leading to 1.49 to 20.09 perplexity reductions as compared to other LAs and even 0.24 reduction over original attention. The effectiveness of our augmented LAs is consistently validated by results on FLASH models and the Wiki40B dataset, demonstrating perplexity reductions ranging from 1.49 to 20.09 as compared to baselines, and even a 0.24 reduction over the original attention.

Table 15. Perplexity evaluation on two tasks: (1) LLaMA models on PG-19 (sequence length is 4K) and (2) FLASH model on Wiki40B (sequence length is 1K).

Model	Loc.	Loc.+Gro.	Loc.**+Conv**		**Augmented LA**
LLaMA-2-7B	21.61	15.04	14.94		**13.47**
LLaMA-2-13B	19.25	12.92	12.92		**11.55**
Model	Loc.	Loc.+Gro.	Gro.	Quad.	**Augmented LA**
FLASH-110M	16.65	16.14	35.25	15.40	**15.16**

H. Augmentation for More Linear Attention Methods

We further extend our augmentation method to four additional types of linear attention methods, including not only FLASH but also the random feature attention (RFA) (Peng et al., 2021), Performer (Choromanski et al., 2021), and Linformer (Wang et al., 2020). Specifically, we evaluated these linear attention methods before and after our augmentation on the decoder-based GPT-2 model and measured the resulting text classification accuracy on the GLUE benchmark (Wang et al., 2018). Tab. 16 demonstrates that our augmentation method consistently improves performance across these methods, achieving non-trivially on average 5.07% $\sim$ 8.05% task accuracy gain. These results validate that our augmentation techniques are generally applicable to different linear attention methods in largely enhancing their achievable performance and efficiency.

Table 16. Augmentation for various linear attention methods.

Method	**SST2**	**RTE**	**MRPC**	**QQP**	**MNLI**	**QNLI**	**WNLI**	**Average**
RFA	83.37	61.01	73.04	82.24	71.73	69.45	45.07	69.42
Aug. RFA	91.28	60.65	75.00	88.53	81.76	69.26	54.93	**74.49**
Performer	86.93	49.46	69.12	76.30	70.60	69.36	38.03	65.69
Aug. Performer	91.51	67.15	72.30	84.61	70.87	63.72	50.70	**71.55**
Linformer	79.47	52.35	68.38	76.30	34.56	69.06	52.11	60.59
Aug. Linformer	92.43	61.37	77.45	88.42	42.63	63.26	54.93	**68.64**