Low-rank Linearization of Large Language Models

Michael Zhang [1] **Aaryan Singhal** [1] **Benjamin Spector** [1] **Simran Arora** [1] **Christopher Ré** [1]

Abstract

Recent subquadratic architectures show exciting progress, now rivaling Transformers in various quality metrics. However, scaling these models up to modern large language model (LLM) sizes can be prohibitively costly. We thus study how to efficiently *linearize* existing LLMs–swapping their attentions with fast analogs, before only adapting the fast analog weights–to quickly create high-quality linear-time and constant memory LLMs. Our approach, Low-rank Linear Conversion via Attention Transfer (LOLCAT), is a simple two-step method that (1) *learns* expressive yet efficient attention analogs by training linear attentions to match the outputs of LLM softmax attentions, before (2) replacing the attentions and adjusting for this swap with only *low-rank adaptation* (LoRA). By linearizing Llama 3 8B and Mistral-7B, LOLCAT produces strong linear attention LLMs that outperform state-of-the-art non-Transformer 7B LLMs, achieving 1.8-4.7 higher points on popular LM Evaluation Harness (LM Eval) tasks, while only training 0.4% of their model parameters with 0.003% of their training tokens. LOLCAT-linearized LLMs further achieve 2.5 - 4× the throughput of original FlashAttention-2 LLMs, while only increasing memory 1.1× when scaling generation length 512× from 512 to 131K tokens. Finally, LOLCAT significantly improves linearization quality and training efficiency, leading to 5.0 higher LM Eval points than concurrent methods with only 0.2% of their training tokens.

1. Introduction

Large language models (LLMs) with billions of parameters and trillions of training tokens have enabled various exciting deep learning capabilities (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022). However, they incur high computational demands in their deployment and development.

Considering deployment, popular state-of-the-art LLMs such as GPT-4 (Achiam et al., 2023), Llama-3-70B (Touvron et al., 2023a;b; AI@Meta, 2024) and Mistral-7B (Jiang et al., 2023) are Transformer-based (Vaswani et al., 2017). They require linearly-growing KV-caches for inference that can bottleneck long-context and large-batch workflows.

Meanwhile, while this motivates *developing* efficient Transformer alternatives (Katharopoulos et al., 2020; Gu et al., 2021; Peng et al., 2023; Poli et al., 2023a; Arora et al., 2024), scaling these models up to modern LLM sizes remains challenging. Despite these architectures' exciting potential for linear time and constant-memory decoding, evaluating their quality traditionally requires training new models from scratch. To perform comparably to Transformer LLMs, such pretraining often entails scaling up to at least 7 billion (7B) parameters and training on 300B to 2 trillion (2T) tokens (Peng et al., 2024; Gu et al., 2021; De et al., 2024). These prohibitive training costs limit efficient model development, with few non-Transformer 7B LLMs, let alone mixture-of-experts (MoE) or 70B models.

We thus study an alternative path to obtaining efficient linear-time LLMs. Instead of pretraining new architectures from scratch, can we *linearize* existing LLMs? By swapping the attention layers of available LLMs with fast attention analogs—*e.g., linear attentions* or recurrent layers—and keeping all other pretrained weights fixed, we can quickly scale up attention analogs at modern LLM sizes.

However, to evaluate effective LLM linearization and motivate linearization over pretraining from scratch, we consider three desiderata. Methods should be: (1) *training-efficient*, requiring only a fraction of the model parameter updates and training tokens typically used to train LLMs from scratch; (2) *quality-preserving*, recovering the zero-shot capabilities of modern Transformer LLMs and at least performing comparably to prior state-of-the-art subquadratic LLMs; and (3) *inference-efficient*, translating the improved time and space complexity of linear models to real-world higher throughput and reduced memory generation in existing LLMs.

Unfortunately, existing linearizing works fall short of all three criteria. Many require full finetuning after attention

[1]Department of Computer Science, Stanford University. Correspondence to: Michael Zhang <mzhang@cs.stanford.edu>.

Proceedings of the 2^{nd} Efficient Systems for Foundation Models Workshop at the International Conference on Machine Learning, Vienna, Austria.

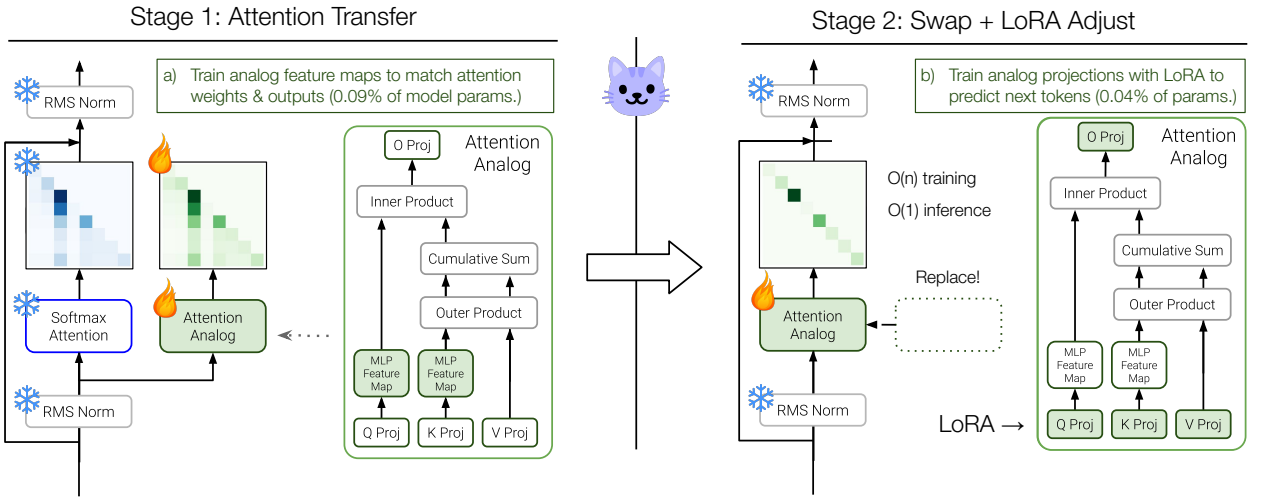


Figure 1. The two stages of LOLCAT. We linearize LLMs into linear versions by (1) training attention analogs to approximate LLM attention layers via attention transfer, and (2) swapping attentions and making minimal additional adjustments via low-rank adaptation.

swapping to recover performance (Kasai et al., 2021; Mao, 2022; Zhang et al., 2024; Mercat et al., 2024). They further often only linearize smaller Transformers (*e.g.,* 110M BERT (Devlin et al., 2018) or 125M GPT-2 (Radford et al., 2019)) for specific tasks (*e.g.,* per-task GLUE classification (Wang et al., 2018)). It thus remains unclear if we can scale linearization effectively to a variety of LLMs.

We thus propose **Lo**w-rank **L**inear **C**onversion with **A**ttention **T**ransfer (LOLCAT), a simple parameter-efficient approach for improved LLM linearization. Like prior work, we first swap each self-attention layer of an existing Transformer with a fast attention analog (*e.g.,* linear attention), before finetuning over some given data to allow the LLM to adjust to the modified layers. However, our key hypothesis is that we can avoid extensive training, yet still recover *LLM* quality, by explicitly *training* attention analogs to approximate the softmax attentions they replace. While intuitive—if we could replace the softmax attentions with perfect linear replicas, then we would not require any additional tuning—prior works suggest that linear attentions either struggle to match softmax expressivity (Keles et al., 2023; Zhang et al., 2024), or are unstable to train when trying to approximate softmax in LLMs (Mercat et al., 2024). We instead show that (1) we can train linear attentions to match softmax attentions in various LLMs, (2) doing so allows low-rank adaptation (LoRA) (Hu et al., 2021) to suffice for adjusting LLMs, and (3) together (1) and (2) enable high-quality linearization with <0.2% of prior methods' training tokens.

In experiments, LOLCAT efficiently enables high quality linearized LLMs. Despite only updating <0.5% of the model parameters with 0.003% of the token budget used in standard pretraining, linearized Mistral-7B and Llama 3 8B outperform strong subquadratic 7B LLMs based on StripedHyena (Poli et al., 2023a), RWKV-v6 (Peng et al., 2024), and Mamba (Gu & Dao, 2023) architectures by 1.8 to 4.7 points (averaged over popular LM Evaluation Harness tasks (Gao et al., 2023)). Mistral-7B-LOLCAT and Llama3-8B-LOLCAT further achieve 2.5 to 4× the throughput (tokens per second) of FlashAttention-2 (Dao, 2023) versions with sublinear memory scaling (incurring 1.1× the GPU memory when generating 512× the tokens, from 512 to 131k token outputs). Finally, LOLCAT significantly improves upon prior linearization methods. We reduce the performance drop incurred with prior approaches by 95.8% (from -37.3 points to -1.6 points) (Kasai et al., 2021), and outperforms concurrent methods by 2.9 to 5.0 LM Eval points with only 0.04% to 0.2% of their training tokens (Mercat et al., 2024).

2. Background: Linear Transformers

Transformers and Attention. Popular LLMs such as Llama 3 (AI@Meta, 2024) and Mistral-7B (Jiang et al., 2023) are built on Transformers. These architectures consist of repeated blocks of *self-attention* layers followed by MLPs, where the attention layers are responsible for sequence modeling (Vaswani et al., 2017). For causal language modeling, attention layers model inputs $\{\boldsymbol{x}_i\}_{i=1}^{l} \in \mathbb{R}^{l \times d}$ (where l is number of tokens, d is head dimension) with query, key, and value weights $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times d}$, and computes attention weights and outputs via

$$\boldsymbol{q}_n = \boldsymbol{x}_n \boldsymbol{W}_q, \boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}_k, \boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}_v \tag{1}$$

$$a_{n,i} = \frac{\exp(\boldsymbol{q}_n^\top \boldsymbol{k}_i / \sqrt{d})}{\sum_{i=1}^{n} \exp(\boldsymbol{q}_n^\top \boldsymbol{k}_i / \sqrt{d})}, \quad \boldsymbol{y}_n = \sum_{i=1}^{n} a_{n,i} \boldsymbol{v}_i \tag{2}$$

for all $n \in [l]$. While expressive, the softmax in Eq. 2 results in requiring all prior $\{\boldsymbol{k}_i, \boldsymbol{v}_i\}_{i \leq n}$ to compute attention output $\boldsymbol{y}_n$. As n grows or we wish to generate many samples

in parallel (*e.g.* in long context or large-batch settings), this growing "KV cache" can incur prohibitive memory costs.

Linear Attention. To get around this, prior work notes that we can perform a similar attention operation, but with only a constant-sized "KV state". This unlocks constant-memory generation that scales independently of sequence length. To see how, note that attention's exp can be viewed as a kernel function $k(\boldsymbol{q}, \boldsymbol{k})$ (Katharopoulos et al., 2020), where in general we can express kernels as the dot product of feature maps $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d'}$. Replacing $\exp(\boldsymbol{q}^\top \boldsymbol{k}/\sqrt{d})$ with $\phi(\boldsymbol{q})^\top \phi(\boldsymbol{k})$ in Eq. 2, we now get *linear attentions*

$$\hat{a}_{n,i} = \frac{\phi(\boldsymbol{q}_n)^\top \phi(\boldsymbol{k}_i)}{\sum_{i=1}^n \phi(\boldsymbol{q}_n)^\top \phi(\boldsymbol{k}_i)}, \quad \hat{\boldsymbol{y}}_n = \sum_{i=1}^n \hat{a}_{i,n} \boldsymbol{v}_i \tag{3}$$

where without the softmax, Eq. 3 can be re-expressed as

$$\hat{\boldsymbol{y}}_n = \frac{\phi(\boldsymbol{q}_n)^\top \left(\sum_{i=1}^n \phi(\boldsymbol{k}_i) \boldsymbol{v}_i^\top \right)}{\phi(\boldsymbol{q}_n)^\top \sum_{i=1}^n \phi(\boldsymbol{k}_i)} = \frac{\phi(\boldsymbol{q}_n)^\top \boldsymbol{s}_n}{\phi(\boldsymbol{q}_n)^\top \boldsymbol{z}_n} \tag{4}$$

for "KV state" $\boldsymbol{s}_n \in \mathbb{R}^{d \times d'}$ and "K state" $\boldsymbol{z}_n \in \mathbb{R}^d$:

$$\begin{aligned} \boldsymbol{s}_n &= \boldsymbol{s}_{n-1} + \phi(\boldsymbol{k}_n) \boldsymbol{v}_n^\top, \quad \boldsymbol{s}_{n-1} = \sum_{i=1}^{n-1} \phi(\boldsymbol{k}_i) \boldsymbol{v}_i^\top \\ \boldsymbol{z}_n &= \boldsymbol{z}_{n-1} + \phi(\boldsymbol{k}_n), \qquad \boldsymbol{z}_{n-1} = \sum_{i=1}^{n-1} \phi(\boldsymbol{k}_i) \end{aligned} \tag{5}$$

Starting with $\boldsymbol{s}_0 = \boldsymbol{0}, \boldsymbol{z}_0 = \boldsymbol{0}$, Eq. 4 and 5 allow computing attention recurrently in $\mathcal{O}(nd'^2)$ time and $\mathcal{O}(d'^2)$ memory for new outputs $\hat{\boldsymbol{y}}_n$. Thus if $d' <$ sequence length l, linear attentions improve Transformer time and space complexity.

Various works propose different functions for ϕ, ranging from those that enforce "valid" attention weights (positive and summing to 1) via $1 + \text{ELU}$ (Katharopoulos et al., 2020) or ReLU (Qin et al., 2022), to those that approximate exp via randomized (Choromanski et al., 2020; Peng et al., 2021; Zheng et al., 2022) or polynomial (Alman & Song, 2023; Keles et al., 2023) features. However, they consistently report underperforming softmax attention in modeling quality.

Rather than design ϕ explicitly, to linearize LLMs we build on prior work that *learns* how to approximate softmax attention, *i.e.,* by parameterizing ϕ as MLPs trained such that $\hat{a}_{n,i} \approx a_{n,i}$ over real data (Zhang et al., 2024). We next describe this featurization and our overall method in greater detail. We further later show that this *attention transfer* enables high-quality linearization in Sec. 4.1.

3. Method: Linearizing LLMs with LOLCATs

We now present LOLCAT, a simple approach to efficiently linearize LLMs. Like prior work (Kasai et al., 2021; Mao, 2022; Zhang et al., 2024), we adopt a two-step procedure, where we swap fast attention analogs into an existing Transformer LLM, before finetuning the parameters to adjust for the modified layers. However, we improve the training efficiency and end-quality of linearized LLMs with three core components: (1) using LoRA instead of full finetuning, (2) training the swapped attention analogs to explicitly approximate softmax attentions, and (3) supporting new learnable attention analogs that improve softmax attention fidelity.

Linearizing with LoRA. As a first step, we simply swap the full-finetuning in existing approaches with low-rank adaptation (LoRA). For every attention layer, we replace the Eq. 2 computation with Eq. 3. For post-swap finetuning, we freeze all layers except for the attention query, key, value, and output projections (the latter acting as a final linear layer to combine outputs in multi-head attention (Vaswani et al., 2017)). To train these layers with LoRA, rather than apply a full-parameter update, *e.g.* $\boldsymbol{W}_q' \leftarrow \boldsymbol{W}_q + \Delta \boldsymbol{W}_q$, we decompose $\Delta \boldsymbol{W}_q$ into the product of two low-rank matrices $\boldsymbol{BA}$, where $\boldsymbol{B} \in \mathbb{R}^{d \times r}$, $\boldsymbol{A} \in \mathbb{R}^{r \times d}$. With $r \ll d$, (*e.g.* 8 versus 128 for Llama-3 and Mistral 7B), LoRA makes linearizing LLMs much more accessible (only updating $<$0.09% of model parameters).

Learning Linear Attentions via Attention Transfer. While prior works show that we can linearize Transformers by simply swapping attention layers before fully finetuning the model on some downstream task, we make linearizing with LoRA sufficient by first explicitly training the attention analogs to approximate their softmax counterparts. We study if the training setup in Zhang et al. (2024) suffices for learning feature maps ϕ that can approximate LLM softmax attentions. In particular, we first parameterize ϕ as an MLP

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x}) \tag{6}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ is a learnable weight matrix, f is a nonlinear activation such as ReLU applied dimension-wise, and $+\boldsymbol{x}$ represents a skip connection. We then learn feature maps ϕ for each head and layer in the LLM. Like Zhang et al. (2024), we note that each $\hat{\boldsymbol{a}}_n$ and $\boldsymbol{a}_n$ represents a valid probability distribution, such that we can simply train ϕ by matching attention weights via a cross-entropy loss $\mathcal{L}_n^{\text{xent}} =$

$$-\sum_{i=1}^n \frac{\exp(\boldsymbol{q}_n^\top \boldsymbol{k}_i/\sqrt{d})}{\sum_{m=1}^i \exp(\boldsymbol{q}_n^\top \boldsymbol{k}_m/\sqrt{d})} \log \frac{\phi(\boldsymbol{q}_n)^\top \phi(\boldsymbol{k}_i)}{\sum_{m=1}^i \phi(\boldsymbol{q}_n)^\top \phi(\boldsymbol{k}_m)} \tag{7}$$

In Sec 4.1, we verify that attention transfer enables successful low-rank linearization for various f, where simply swapping then finetuning can fail to produce viable LLMs.

Expanding Learnable ϕ Parameterizations. We evaluate two specific ϕ parameterizations for the LLM regime. First, we consider a pure linear attention setup as in Eq. 6. In Sec. 4.1, we explore how various activations f used in prior linearizing works transfer to the LLM setting.

However, to further improve linearization quality, we also propose a hybrid linear attention + sliding window parameterization. Motivated by prior works that show quality improvements when combining attentions layers with more efficient convolutions (Fu et al., 2022), recurrences (De et al., 2024), or linear attentions (Arora et al., 2024; Munkhdalai et al., 2024), we combine short sliding windows of softmax attention (Beltagy et al., 2020; Zhu et al., 2021) with linear attention in a single learnable feature map. For token n in a sequence, we compute attentions $[\hat{a}_{n,1} \dots \hat{a}_{n-w}, a_{n,n-w+1}, \dots, a_{n,n}]$ and output $\tilde{\boldsymbol{y}}_n$ via

$$\tilde{\boldsymbol{y}}_n = \sigma \sum_{i=n-w+1}^{n} a_{n,i}\boldsymbol{v}_i + (1-\sigma)\sum_{i=1}^{n-w} \hat{a}_{n,i}\boldsymbol{v}_i \quad (8)$$

where $a_{n,i}$ are softmax attention weights defined over short sliding window of size w, $\hat{a}_{n,i}$ are linear attention weights, and σ is a learnable interpolating factor.

4. Experiments

4.1. Low-rank Linearization via Attention Transfer

We first show that we can learn softmax-approximating linear attentions in LLMs via attention transfer. Furthermore, doing so often enables successful low-rank linearization where simply swapping linear attentions before LoRA-finetuning fails. In Table 1, we evaluate how various prior ϕ perform when used to linearize base Llama 3 8B LLMs with the Alpaca dataset (Taori et al., 2023). We use Alpaca due to its ability to showcase general instruction-following in 7B LLMs despite its relatively small size (50k samples, or 10M tokens). For ϕ, we compare the $1 + \text{ELU}$ (Katharopoulos et al., 2020), ReLU (Kasai et al., 2021), and exp (Zhang et al., 2024) feature map activations in prior work, and evaluate the effect of first training the ϕ to match original softmax attention layers (Transfer? ✓).

In all cases, training linear attentions to match softmax as in (Zhang et al., 2024) improves linearized LLM perplexity on validation responses. Softmax approximation quality of the linear attentions further emerges as a key quality indicator for low-rank linearization. We further show in Fig. 3 through Fig. 6 how attention transfer enables recovery of softmax attention weights in various linear attentions feature maps, while no attention transfer results in poor attention weight fidelity.

4.2. LOLCATs for State-of-the-Art Subquadratic Quality

To further test whether LOLCAT enables effective LLM linearization, we next test the zero-shot language modeling quality of LLMs linearized with LOLCAT. To see if we can use a small amount of unrelated data to recover zero-shot capabilities, we linearize Mistral-7B-v0.1 and Llama 3 8B models with the Alpaca dataset, and test on five popular zero-shot LM Evaluation Harness tasks (Gao et al., 2023).

Table 1. Perplexity and attention approximation quality strongly correspond for low-rank linearization. PPL@0,@2 reported after swapping, 2 finetuning epochs. MSE* multiplied by 1e3.

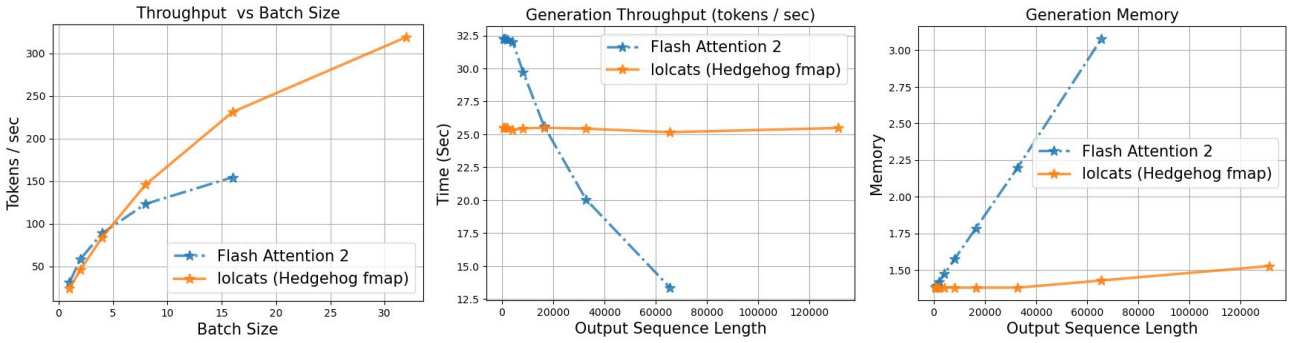
Attention		Llama 3 8B			
Feature Map	Transfer?	PPL@0	PPL@2	MSE*	KLD
$1 + \text{ELU}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x})$	✗	6.58e4	555.99	17.13	3.20
$\text{ReLU}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x})$	✗	6.20e4	1420.72	17.25	3.31
$\exp(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x})$	✗	1.02e4	6.78	11.88	1.76
$1 + \text{ELU}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x})$	✓	386.92	3.38	1.89	0.28
$\text{ReLU}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x})$	✓	414.18	3.52	1.76	0.21
$\exp(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{x})$	✓	10.42	**3.11**	**0.98**	**0.12**
Softmax		20.51	2.15	0.00	0.00

Table 2. LM Evaluation Harness, zero-shot default setting.

Linearized	Tokens (B)	PIQA (acc.)	ARC-e (acc.)	ARC-c (acc. norm)	HellaSwag (acc. norm)	WinoGrande (acc.)
Mistral 7B T2R	**+0.04**	53.5	27.4	26.8	26.6	51.2
Mistral 7B SUPRA	+20	80.1	74.6	42.3	74.8	67.4
Mistral 7B SUPRA	+100	80.4	75.9	45.8	77.1	70.3
Llama 3 8B LOLCAT-l	**+0.04**	78.4	74.8	42.7	67.8	53.9
Llama 3 8B LOLCAT-w	**+0.04**	80.3	80.1	51.1	76.7	73.2
Mistral 7B LOLCAT-l	**+0.04**	79.4	77.5	45.6	73.5	53.1
Mistral 7B LOLCAT-w	**+0.04**	**81.1**	**81.4**	**53.6**	**77.4**	**70.6**
Subquadratic						
StripedHyena	–	78.8	77.2	40.0	76.4	–
RWKV-v5 (EagleX)	2250	77.6	74.5	41.6	–	73.3
Mamba-7B	1200	81.0	77.5	46.7	77.9	71.8
RWKV-v6 (Finch)	1420	78.7	76.8	46.3	75.1	70.0
Transformer						
Llama 2 7B	2000	78.0	76.2	46.3	75.6	69.1
Mistral 7B	8000 (?)	82.1	80.9	53.8	81.0	74.1
Llama 3 8B	15000	79.5	80.0	53.4	79.1	70.0

From the best results in Sec. 4.1, we use the exp feature map for LOLCAT, and test both vanilla linear attention (LOLCAT-l) and hybrid short window analogs (LOLCAT-w; size 16 windows). We train the analog MLPs for two epochs with learning rate 1e-2 before swapping, and LoRA-finetuning the LLMs for two epochs with learning rate 1e-4. For all stages we use AdamW optimizer and batch size 8 with gradient accumulation.

In Table 2, we find LOLCAT can achieve state-of-the-art subquadratic LLM quality with minimal training resources (*e.g.*, 40M tokens, 0.003% of the 1.5T tokens used to pretrain prior models (Peng et al., 2023)). In particular, LOLCAT produces linear LLMs that outperform strong 7B LLMs such as RWKV-v6 Finch World v2.1 (Peng et al., 2024), StripedHyena-Nous-7B (Poli et al., 2023b), and Mamba 7B (Gu & Dao, 2023; Mercat et al., 2024). Furthermore, to shed light on LOLCAT's attention transfer, we compare against two available linearization methods for Mistral-7B that swap attention analogs but do not train these layers to match softmax attention: Transformer-to-RNN (T2R) (Kasai et al., 2021) and SUPRA (Mercat et al., 2024) approaches. LOLCAT improves performance by 35.7 points over T2R, while also outperforming the SUPRA approaches



(a) (b) (c)

Figure 2. Inference scaling performance of LOLCAT. (Hedgehog fmap) denotes the trainable exp feature map used in Zhang et al. (2024).

by 2.9 and 5.0 points, despite only using 0.04% to 0.2% of their training token budgets and finetuning with LoRA.

4.3. Linearizing for Subquadratic Efficiency

Finally, we evaluate LOLCAT inference and generation efficiency. Via linear attention, we find we can scale dramatically better than existing Transformers with FlashAttention-2 (Dao, 2023) over longer generations and larger batch size. We benchmark LOLCAT-l and FlashAttention-2 versions of Mistral-7B in full bfloat16 precision on one 40GB A100.

In Fig. 2(a) we report the effect of scaling batch size on throughput, with a fixed 2048 token prompt and 128 token generation. We find that for small batches, FlashAttention-2 slightly outperforms linear attention. However, as batch size scales, LOLCAT becomes significantly faster than attention. Furthermore, the memory consumption of FlashAttention vastly outpaces linear attention; as a result, it cannot run in batches beyond 16 due to memory limits.

In Fig. 2(b, c), we show how LOLCAT scales with generation length. We consider the setting of single-batch inference with a short prompt (128 tokens) and varying generation (up to 131k tokens). Below 16k tokens, FlashAttention-2 runs more quickly, but as sequence length increases, linear attention provides up to twice the throughput. Beyond sequences of 64k, FlashAttention-2 runs out of memory. However, we observe that, due to the constant memory and the recurrent nature of linearized LLMs, LOLCAT maintains nearly constant throughput and memory usage as we increase the number of tokens generated from 512 to 131K tokens.

5. Conclusion

In this work, we propose LOLCAT, an efficient LLM linearization approach that (1) trains attention analogs to approximate an LLM's self-attentions, before (2) swapping the attentions and only finetuning the replacing attentions with LoRA. On popular zero-shot evaluation benchmarks, we find this enables producing high-quality high-inference efficiency LLMs that outperform prior Transformer alternatives, while only updating 0.1% of model parameters and requiring 0.003% of the training tokens to achieve similar quality with LLM pretraining. Our findings also contribute to improving linearization methods; we evaluate existing and concurrent approaches and find LOLCAT leads to 5.0 point improvements over concurrent linearizing methods with only 0.04% of their linearizing tokens.

References

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

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Alman, J. and Song, Z. Fast attention requires bounded entries. *arXiv preprint arXiv:2302.13214*, 2023.

Arora, S., Eyuboglu, S., Zhang, M., Timalsina, A., Alberti, S., Zinsley, D., Zou, J., Rudra, A., and Ré, C. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 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. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

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

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., Haroun, R., Berrada, L., Chen, Y., Srinivasan, S., et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

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.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

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

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

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

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

Kasai, J., Peng, H., Zhang, Y., Yogatama, D., Ilharco, G., Pappas, N., Mao, Y., Chen, W., and Smith, N. A. Finetuning pretrained transformers into RNNs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10630–10643, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.830. URL https://aclanthology.org/2021.emnlp-main.830.

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

Keles, F. D., Wijewardena, P. M., and Hegde, C. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pp. 597–619. PMLR, 2023.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Mao, H. H. Fine-tuning pre-trained transformers into decaying fast weights. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10236–10242, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.697. URL https://aclanthology.org/2022.emnlp-main.697.

Mercat, J., Vasiljevic, I., Keh, S., Arora, K., Dave, A., Gaidon, A., and Kollar, T. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

Munkhdalai, T., Faruqui, M., and Gopal, S. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

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

Peng, B., Goldstein, D., Anthony, Q., Albalak, A., Alcaide, E., Biderman, S., Cheah, E., Ferdinan, T., Hou, H., Kazienko, P., et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

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

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023a.

Poli, M., Wang, J., Massaroli, S., Quesnelle, J., Carlow, R., Nguyen, E., and Thomas, A. StripedHyena: Moving Beyond Transformers with Hybrid Signal Processing Models, 12 2023b. URL https://github.com/togethercomputer/stripedhyena.

Qin, Z., Han, X., Sun, W., Li, D., Kong, L., Barnes, N., and Zhong, Y. The devil in linear transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 7025–7041, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.473. URL https://aclanthology.org/2022.emnlp-main.473.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

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. https://github.com/tatsu-lab/stanford_alpaca, 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.

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

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.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Zhang, M., Bhatia, K., Kumbong, H., and Re, C. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4g02l2N2Nx.

Zheng, L., Wang, C., and Kong, L. Linear complexity randomized self-attention mechanism. In *International conference on machine learning*, pp. 27011–27041. PMLR, 2022.

Zhu, C., Ping, W., Xiao, C., Shoeybi, M., Goldstein, T., Anandkumar, A., and Catanzaro, B. Long-short transformer: Efficient transformers for language and vision. *Advances in neural information processing systems*, 34: 17723–17736, 2021.

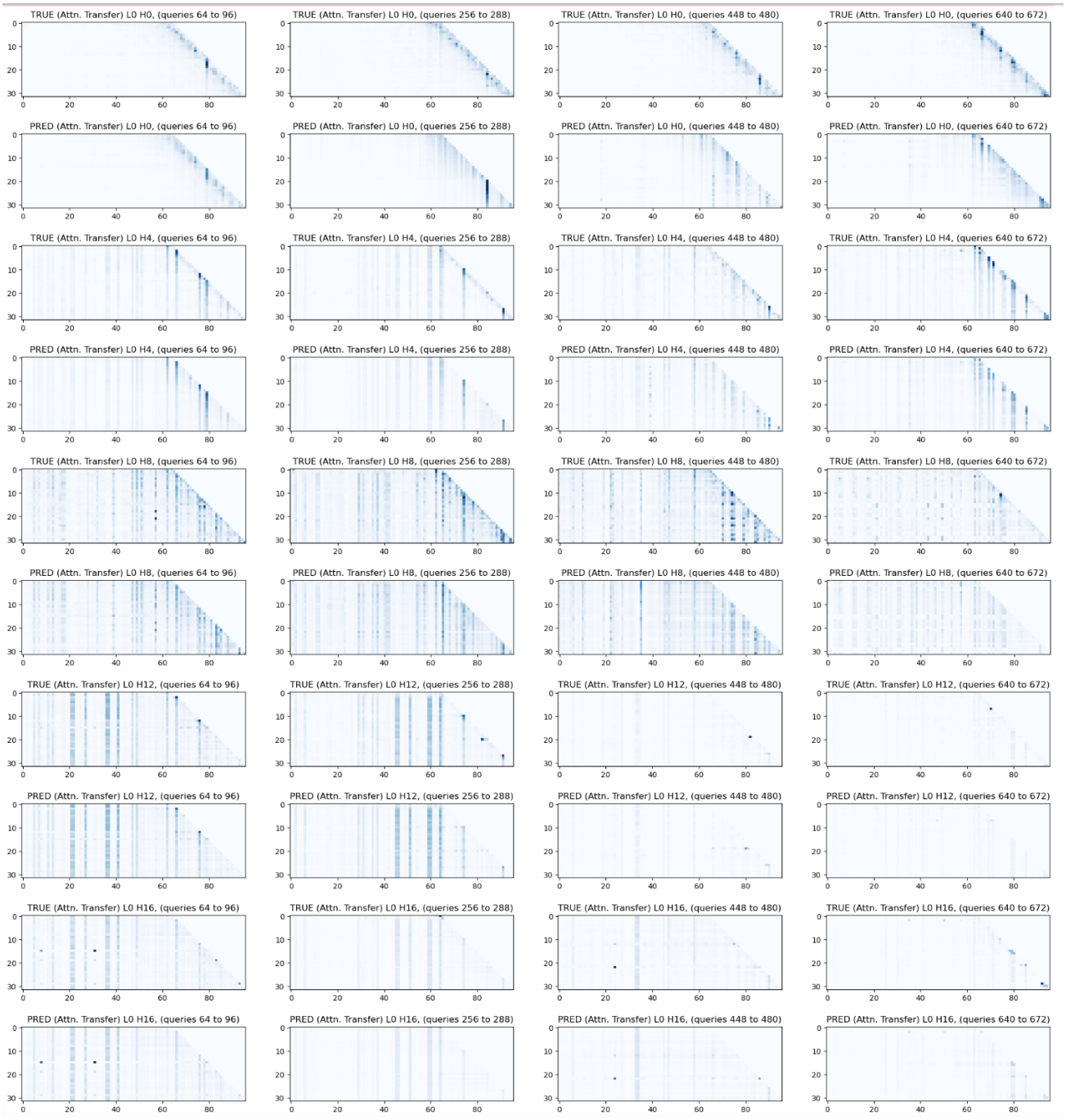


Figure 3. Linear attention (PRED) and softmax attention (TRUE) weights for exp learned feature map, with attention transfer.

Figure 4. Linear attention (PRED) and softmax attention (TRUE) weights for exp learned feature map, without attention transfer.

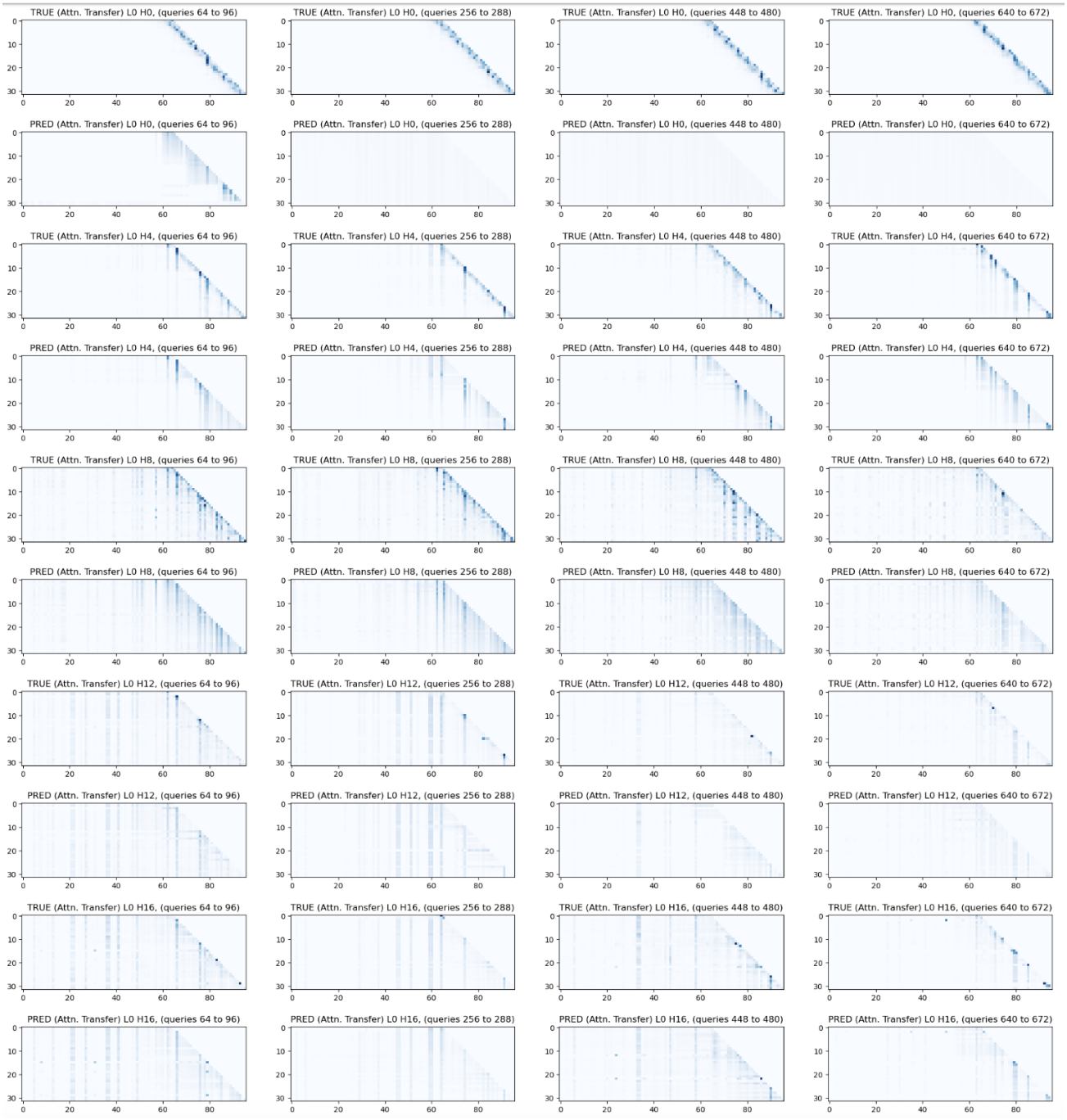


Figure 5. Linear attention (PRED) and softmax attention (TRUE) weights for ReLU learned feature map, with attention transfer.

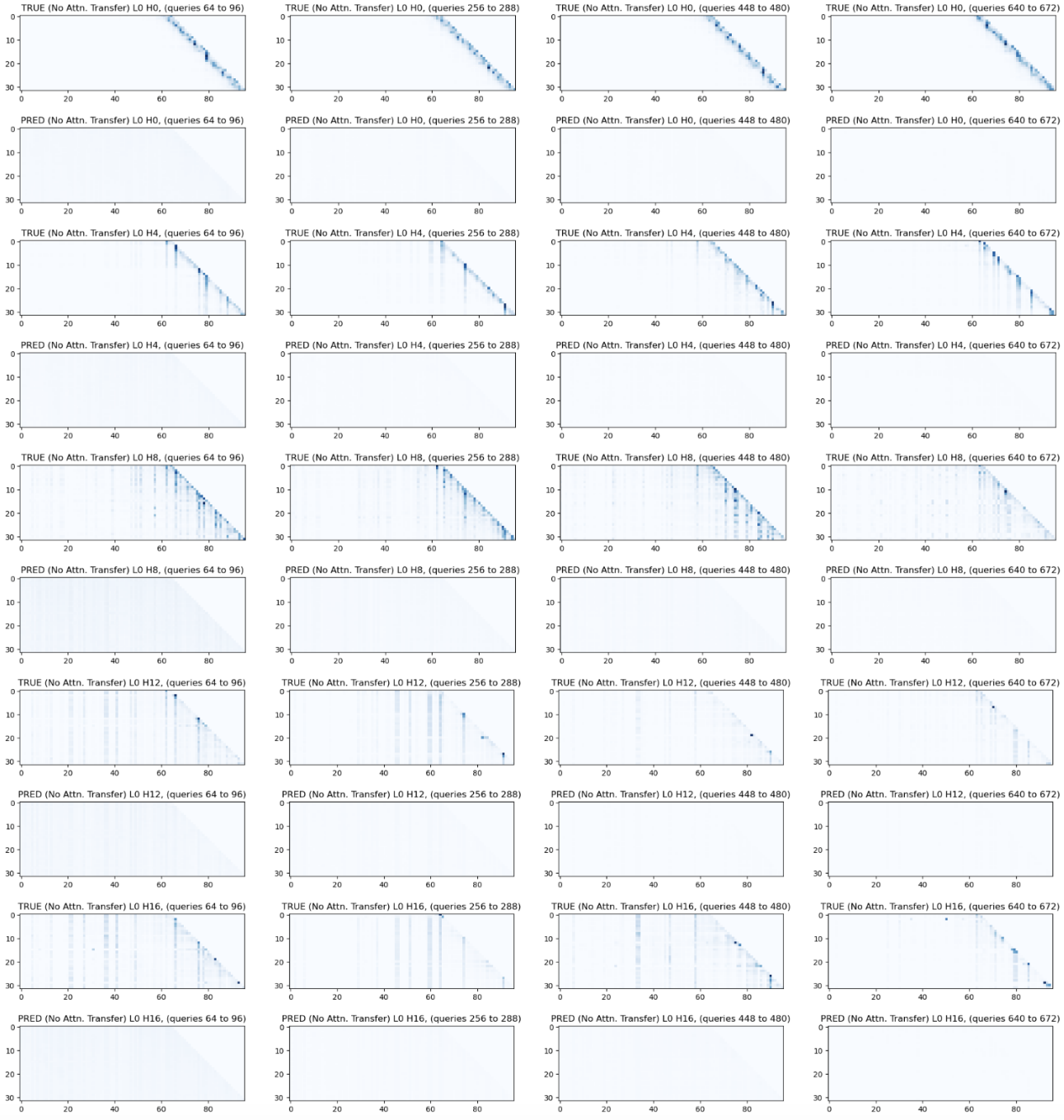


Figure 6. Linear attention (PRED) and softmax attention (TRUE) weights for ReLU learned feature map, without attention transfer.