MIXTURE OF CONVOLUTIONS: TOKEN-ADAPTIVE SHORT CONVOLUTION FOR SEQUENCE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Short convolution has become a common local mixing component in modern sequence models, yet existing variants are typically static: a single learned kernel defines the same local aggregation rule at every position. We introduce Mixture of Convolutions (MoC), a simple adaptive generalization of short convolution that replaces this single kernel with a mixture over learned convolutional bases. Evaluated in both Transformer and Gated DeltaNet architectures, MoC shows broad improvements in language-modeling perplexity over static short convolution, at modest additional cost. We additionally study how short convolutions are optimized, and find that their performance is sensitive to the interaction between initialization scale and learning rate, in a way that affects static short convolution as well as MoC. These results suggest that token-adaptive local mixing is a simple and effective extension of a widely used sequence-modeling primitive.

1 INTRODUCTION

Short depthwise convolution is now a small but important component of many sequence-modeling architectures. Primer showed that inserting short convolutions into Transformer attention projections can improve language modeling (So et al., 2021), and related causal depthwise convolutions have since become common in efficient recurrent and linear-attention sequence models (Dao & Gu, 2024; Yang et al., 2025; Team et al., 2025). Although these kernels are extremely local, usually spanning only a few tokens, they often provide consistent gains at modest cost relative to the rest of the block.

Despite their usefulness, these short convolutions are typically static. A single learned depthwise kernel is shared across all positions, so the same local aggregation rule is applied regardless of the current token or context. This is a potentially restrictive parameterization: the surrounding sequence representations vary across positions, but the local operator used to combine them does not.

We propose Mixture of Convolutions (MoC), a simple adaptive generalization of short convolution. Instead of learning one kernel, MoC learns a small set of convolutional bases and constructs the effective kernel at each position as a softmax-normalized mixture of these bases. The operator remains causal, depthwise, and local, and standard static short convolution is recovered as the single-basis case.

Concurrent work independently explores a closely related form of dynamic short convolution and demonstrates the broader utility of input-dependent convolution across multiple architectures and model scales (Sieberling et al., 2026). We compare these parameterizations directly under matched experimental conditions.

We additionally study how short convolutions are optimized. We find that their performance is sensitive to the learning rate relative to the initialization scale, and that the learning rates typically applied to these parameters are poorly matched to standard fan-in initialization, for both static short convolution and the operator we propose.

We evaluate MoC in Transformer (Vaswani et al., 2017) and Gated DeltaNet (Yang et al., 2025) models, where it consistently reduces perplexity on FineWeb-Edu and WikiText-103 in all evaluated settings relative to static short convolution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 SHORT CAUSAL DEPTHWISE CONVOLUTION

We use *short convolution* to refer to a causal depthwise convolution with a small kernel size applied along the sequence dimension, following the channelwise convolution used in depthwise-separable architectures (Chollet, 2017). Given a sequence representation $x_{1:T}$, a short convolution computes

$$y_t = \sum_{s=0}^{S-1} w_s \odot x_{t-s},$$

where S is the kernel size, $w_s \in \mathbb{R}^d$ is the depthwise kernel at offset s, and $\odot$ denotes elementwise multiplication. In the standard formulation, the same depthwise kernel is shared across all sequence positions.

3 MIXTURE OF CONVOLUTIONS

Mixture of Convolutions generalizes short convolution by replacing the single depthwise kernel with a set of learned convolutional bases. At each position, these bases are combined into an effective kernel and applied over the same causal window as standard short convolution.

Let $B \in \mathbb{R}^{k \times S \times d}$ denote a bank of k depthwise convolutional bases, where S is the kernel size and d is the channel dimension. We refer to the individual bases B_i as experts and use the two terms interchangeably, following the mixture-of-experts formulation of Jacobs et al. (1991), in which a gate produces a normalized distribution over a set of experts. Given a routing input z_t, MoC computes dense mixture weights

$$\alpha_t = \mathrm{softmax}(G(z_t)) \in \mathbb{R}^k,$$

where G is a router. The token-specific depthwise kernel is then

$$w_{t,s} = \sum_{i=1}^{k} \alpha_{t,i} B_{i,s},$$

and the output is computed as

$$y_t = \sum_{s=0}^{S-1} w_{t,s} \odot x_{t-s}.$$

When $k = 1$, the mixture contains a single basis and MoC reduces exactly to standard short convolution. For $k > 1$, the local aggregation rule can vary across positions while retaining the same depthwise and fixed-window structure.

The basis-mixture parameterization provides a controlled form of adaptivity. Rather than generating an unconstrained convolutional kernel at each position, MoC restricts each kernel to a positive combination of shared learned bases, with the standard softmax formulation yielding a convex mixture.

4 EXPERIMENTS

We evaluate MoC in both Transformer and Gated DeltaNet models. We compare against static short convolution and the low-rank dynamic-convolution parameterization (LRDC) introduced in concurrent work (Sieberling et al., 2026), while keeping the surrounding architecture and training setup fixed.

4.1 EXPERIMENTAL SETUP

We compare four configurations across two model sizes:

- **No short convolution**: the local convolution operator is removed.
- **Static short convolution**: standard causal depthwise convolution with a single learned kernel.

- **LRDC**: low-rank dynamic convolution with rank $R = 16$ (Sieberling et al., 2026).
- **MoC**: Mixture of Convolutions with $k = 16$ learned convolutional bases.

Unless otherwise stated, all short convolutions use a kernel size of $S = 4$. For LRDC and MoC, we set $R = 16$ and $k = 16$, respectively, resulting in approximately equal parameter counts.

For MoC, the mixture weights are produced by applying a linear projection to the normalized block input hidden state, followed by a softmax over the k bases. The router projection is initialized to zero, yielding an initially uniform basis mixture. The static short-convolution kernels and MoC basis kernels are initialized from $\mathcal{U}(-0.002\sqrt{3},\ 0.002\sqrt{3})$, corresponding to a standard deviation of 0.002. LRDC follows the parameterization and initialization described by Sieberling et al. (2026). In all convolutional variants, the operator is applied to the projected query, key, and value representations, each with its own parameters.

We train all models on FineWeb-Edu (Penedo et al., 2024). The smaller models are trained for 5B tokens and the larger models for 15B tokens. All runs use a context length of 2048, a global batch size of 524,288 tokens, and the Llama 3 tokenizer (Grattafiori et al., 2024).

We evaluate language modeling on the FineWeb-Edu validation set, WikiText-103 (Merity et al., 2017), and LAMBADA (Paperno et al., 2016). We additionally report zero-shot performance using the LM Evaluation Harness (Gao et al., 2023).

The no-convolution Transformer models contain 190.45M and 458.68M parameters, while the corresponding Gated DeltaNet models contain 190.41M and 458.48M parameters. Static short convolution adds 0.11M and 0.29M parameters to the Transformer models and 0.08M and 0.20M parameters to the Gated DeltaNet models. MoC adds 2.21M and 5.89M parameters to the Transformer models and 1.63M and 4.33M parameters to the Gated DeltaNet models.

At the 190M scale, we additionally evaluate a parameter-matched static-convolution baseline, which allocates the additional MoC parameters to the SwiGLU expansion instead of the convolution operator. The Transformer expansion ratio is increased from $3.0\times$ to $3.1\times$, and the Gated DeltaNet expansion ratio from $3.0\times$ to $3.075\times$, yielding approximately the same parameter count as MoC at this scale.

Our Transformer models use query-key normalization (Henry et al., 2020) together with a normalized and gated attention output: we apply per-head RMS normalization and the per-head sigmoid gate of Qiu et al. (2025) to the attention output before the output projection. This mirrors the use of post-sequence-mixing normalization and output gating in standard Gated DeltaNet blocks. We apply this attention block to all Transformer variants.

We optimize dense projection parameters with NorMuon (Li et al., 2025) using a learning rate of 10^{-2}, and optimize all remaining parameters with AdamW (Loshchilov & Hutter, 2019) using a learning rate of 10^{-3}. We use weight decay 0.1, cosine learning-rate decay to 0 with a 2% warmup, and apply the same optimization settings across all convolutional variants.

4.2 Main Results

Table 1 reports our main results. Adding static short convolution improves language-modeling perplexity in nearly all settings. At the 190M scale, increasing the MLP width to match the parameter count of MoC yields only modest additional gains, indicating that the improvements from dynamic convolution are not explained simply by parameter count. Among the dynamic variants, MoC obtains lower perplexity than LRDC across all reported language-modeling evaluations, with small but consistent margins. We further compare the two parameterizations in Appendix B.

Examining the language-modeling results in cross-entropy space, the improvement from adding static convolution is smaller at the larger scale on both FineWeb-Edu and WikiText-103, in both architectures. The additional improvement from replacing static convolution with MoC does not shrink comparably. It remains unclear whether this behavior extends beyond the model sizes and training budgets evaluated here.

Zero-shot downstream results show a weaker but broadly compatible pattern. Across all four model and scale settings, the no-convolution baseline has the lowest average accuracy among the configurations shown, suggesting that the language-modeling improvements from local mixing are not

Table 1: Main results comparing no short convolution, static short convolution, low-rank dynamic convolution (LRDC), and MoC. Static + (MLP) denotes a parameter-matched static short convolution baseline with the additional MoC parameters allocated to the MLP expansion. Best results within each model and scale are shown in bold, and second-best results are underlined. The average is computed over the eight accuracy tasks.

Config	F-Edu ppl↓	Wiki. ppl↓	LAMB. ppl↓	LAMB. acc↑	Hella. acc-n↑	PIQA acc↑	Wino. acc↑	ARC-e acc↑	ARC-c acc-n↑	SciQ acc↑	Avg. acc↑
Transformer, 190M parameters, 5B tokens											
No conv.	19.17	36.07	60.46	30.33	34.71	64.53	49.41	55.39	25.85	81.40	48.80
Static conv.	18.91	35.06	55.77	30.35	35.04	64.80	51.07	55.05	**26.54**	81.90	49.25
Static + (MLP)	18.87	34.77	57.19	29.94	35.14	64.91	50.43	55.93	25.94	**82.90**	**49.31**
LRDC	18.87	34.91	55.32	29.56	35.01	**64.96**	**51.22**	**56.65**	**26.54**	80.50	49.21
MoC	**18.72**	**34.73**	**53.42**	**30.80**	**35.23**	64.53	50.43	55.68	25.77	82.10	49.22
Transformer, 460M parameters, 15B tokens											
No conv.	13.88	23.52	22.34	39.28	47.27	69.29	54.62	65.11	32.00	86.40	56.28
Static conv.	13.82	23.29	22.45	**40.07**	47.61	69.10	**54.78**	**66.41**	32.25	87.00	56.75
LRDC	13.72	23.10	22.07	39.36	47.95	**70.08**	54.70	66.29	**32.34**	**87.10**	**56.83**
MoC	**13.67**	**22.98**	**21.26**	39.71	**48.10**	69.59	54.54	66.25	32.08	86.80	56.72
Gated DeltaNet, 190M parameters, 5B tokens											
No conv.	19.89	39.24	72.11	26.10	33.78	64.20	51.93	54.34	25.16	77.30	47.54
Static conv.	19.45	37.81	**63.48**	27.34	**34.93**	64.53	**52.25**	**56.27**	26.19	**80.90**	**48.92**
Static + (MLP)	19.44	37.19	69.20	26.06	34.76	**65.18**	50.12	55.30	25.60	80.30	48.19
LRDC	19.29	36.66	70.82	25.62	34.41	64.15	52.17	55.93	**26.79**	79.70	48.40
MoC	**19.19**	**36.58**	65.01	**28.35**	34.92	64.85	50.12	56.10	**26.79**	80.40	48.79
Gated DeltaNet, 460M parameters, 15B tokens											
No conv.	14.23	24.90	24.70	37.24	45.94	67.90	53.12	63.85	30.03	86.10	54.88
Static conv.	14.07	24.53	22.60	**39.43**	46.71	**69.91**	53.67	67.00	31.06	86.20	56.28
LRDC	13.95	24.00	22.32	38.93	**47.40**	69.10	53.35	65.24	30.12	86.30	55.78
MoC	**13.89**	**23.81**	**21.80**	38.73	47.36	69.42	**54.78**	**67.17**	**31.74**	**86.90**	**56.59**

confined to perplexity alone. However, the differences among static convolution, LRDC, and MoC are relatively small and inconsistent across tasks. We therefore treat the downstream results as evidence that short convolution is generally helpful in these settings, but not as a reliable basis for ranking the convolution parameterizations.

4.3 Short-Convolution Optimization

Short-convolution optimization is sensitive to initialization scale. Under the optimization settings used in our main experiments, substantially smaller initialization scales consistently improve both static short convolution and MoC. We next investigate this behavior and find that the results are consistent with a more general dependence on the optimizer step size relative to the initial parameter scale.

We adapt the standard fan-in initialization for short convolution to MoC by multiplying the per-basis scale by $\sqrt{k}$, compensating for the variance reduction from averaging the k bases under the initially uniform router. The resulting per-basis initialization is

$$\mathcal{U}\left(-\sqrt{\frac{k}{S}}, \sqrt{\frac{k}{S}}\right),$$

with standard deviation $\sqrt{k/(3S)}$. For the kernel size $S = 4$ used in our main experiments, this corresponds to approximately 0.29 for static short convolution and 1.15 for MoC with $k = 16$ bases. We evaluate these fan-in initializations together with smaller standard deviations of 0.02 and 0.002.

To maintain representation scale across different initializations, we apply query-key normalization (Henry et al., 2020) and retrieved value normalization, implemented as per-head RMS normalization of the attention output. This normalization is already part of standard Gated DeltaNet blocks (Yang et al., 2025). We pair it with per-head gated attention (Qiu et al., 2025), which retains input-dependent control over the retrieved value contribution to the residual stream. Under this normalization scheme, smaller initialization consistently improves both static convolution and MoC.

Table 2: Effect of convolutional initialization scale. Query-key normalization and retrieved value normalization are used in every configuration. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Init. std.	Static convolution	MoC
1.15	25.49	25.32
0.29	24.99	24.04
0.02	24.41	23.97
0.002	24.21	23.79

We hypothesize that this improvement is caused by larger optimizer updates relative to the initial parameter scale. Under AdamW, reducing the initialization scale while holding the learning rate fixed increases this initial update-to-parameter ratio.

We test this hypothesis using the fan-in initialization for each operator. For comparison, we include the corresponding small-initialization result from Table 2. We compare the convolution learning rate used in our main experiments, 10^{-3}, against a convolution-specific learning rate scaled by the ratio between the fan-in and small initialization scales:

$$\eta_{\text{matched}} = \eta_{\text{small}} \frac{\sigma_{\text{fan-in}}}{\sigma_{\text{small}}}.$$

The matched learning rate is applied only to the convolutional parameters, and their AdamW weight decay is adjusted to preserve the same per-step multiplicative shrinkage. This scaling is intended to approximately preserve the initial optimizer-induced movement relative to the parameter scale; it is not intended to identify the validation-optimal convolution learning rate. Appendix C.1 provides a coarse learning-rate sweep at the fan-in initialization.

Table 3: Interaction between convolution initialization scale and learning rate. Each large learning rate is obtained by matching the initial update-to-parameter ratio of the corresponding small-initialization configuration, giving $\eta = 0.14$ for static convolution and $\eta = 0.58$ for MoC. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Operator	Init. std.	LR $= 10^{-3}$	Matched LR
Static	0.29	24.99	24.14
	0.002	24.21	24.18
MoC	1.15	25.32	23.83
	0.002	23.79	23.87

At fan-in initialization, using the convolution learning rate from our main experiments yields substantially worse performance for both operators. Increasing the learning rate in proportion to the initial parameter scale yields performance similar to the corresponding small-initialization configurations. Applying the same large learning rates to the small-initialization configurations does not provide a comparable improvement. Together, these results are consistent with performance depending on the optimizer update relative to the convolutional parameter scale, rather than on either small parameter values or large learning rates in isolation. This analysis is specific to AdamW; optimizers that scale updates relative to parameter magnitude may exhibit different sensitivity to initialization scale.

4.4 RESIDUAL CONNECTIONS

Prior work often applies short convolution with an explicit residual connection (Allen-Zhu, 2025; Tian et al., 2026). Motivated by the dependence on optimizer movement relative to the initial parameter scale identified in Section 4.3, we investigate whether the effect of this residual changes across optimization regimes. We compare static short convolution with and without an explicit residual connection under fan-in and small initialization in Table 4.

Table 4: Interaction between static short-convolution initialization and an explicit residual connection. All configurations use a convolution learning rate of 10^{-3}. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Init. std.	No residual	Residual
0.29	24.99	24.56
0.002	24.21	24.55

With fan-in initialization, the explicit residual connection improves performance, whereas under small initialization it degrades performance. At the learning rate used here, fan-in initialization produces much smaller optimizer updates relative to the convolutional parameter scale, so the local operator adapts more slowly. In this regime, the residual provides a reliable current-token path while the convolutional parameters evolve. Under small initialization, the convolution can adapt its output much more rapidly, and this residual anchor is no longer beneficial.

The degradation under small initialization may also reflect an imbalance in adaptation and in output contribution across the kernel. The past-position taps remain small and can change substantially relative to their parameter scale, but their effect on the convolution output is diluted by the large fixed current-token contribution introduced by the residual. The current-position path therefore remains dominant even as the contextual taps adapt.

A separate consideration is the role of the residual in maintaining forward scale. When normalization is not used, an explicit residual can help correct the reduced output scale induced by small convolutional initialization. This may change the effect of the residual in that setting, but it is distinct from the relative-update mechanism studied here. We also emphasize that these conclusions are specific to the post-QKV convolution placement evaluated in this work; other placements may exhibit different optimization trade-offs.

More generally, this suggests that design choices surrounding short convolution may not transfer across initialization regimes, since components that are beneficial under fan-in initialization can be unnecessary or harmful once the convolution adapts more rapidly. Appendix C.4 reports a similar interaction with the post-convolution nonlinearity used in Gated DeltaNet.

4.5 BASIS COUNT AND KERNEL SIZE

Table 5 studies the effect of the number of convolutional bases and the convolution kernel size.

Table 5: Effect of the number of convolutional bases and convolution kernel size on Transformer models trained on 2.5B tokens. The basis-count sweep uses $S = 4$, with $k = 0$ denoting no short convolution and $k = 1$ reducing to static short convolution. The kernel-size sweep uses $k = 16$.

k	Params. (M)	F-Edu ppl↓
0	106.5	24.78
1	106.6	24.21
2	106.7	24.02
4	106.9	23.97
8	107.2	23.85
16	108.0	23.79
32	109.4	23.73
64	112.4	23.61
128	118.3	**23.50**

S	Params. (M)	F-Edu ppl↓
1	107.1	24.95
2	107.4	24.19
3	107.7	23.89
4	108.0	23.79
5	108.3	23.77
6	108.6	**23.74**

Performance improves monotonically as additional bases are introduced. Increasing the kernel size also improves performance, with gains diminishing beyond the commonly used setting of $S = 4$.

4.6 Efficiency

MoC adds computation beyond static short convolution because each token mixes multiple convolutional bases. To measure the practical cost of this operator, we benchmark the 460M-parameter Transformer used in our experiments. Throughput is normalized to an otherwise identical model without short convolution. Static short convolution uses the implementation from Flash Linear Attention (Yang & Zhang, 2024).

Table 6: Relative throughput of a 460M-parameter Transformer normalized to the same model without short convolution. Benchmarks use a sequence length of 2048 and a per-device batch size of 3, and are run on an NVIDIA RTX 5090 GPU with kernel size $S = 4$ and $k = 16$ MoC experts.

Benchmark	No conv.	Static conv.	MoC
Forward only	$1.000\times$	$0.977\times$	$0.958\times$
Forward + backward	$1.000\times$	$0.949\times$	$0.910\times$

Our implementation uses fused Triton kernels for MoC to avoid materializing the dynamic convolution kernel. The throughput reported here uses softmax routing, and further optimization remains possible. For the query and key convolutions, the subsequent query-key normalization makes the softmax denominator redundant, allowing unnormalized exponential routing and fusion of the exponential into the router projection epilogue. The small router projections may also be more efficiently combined with one another. In addition, the post-convolution query-key normalization and RoPE application could potentially be fused into the convolution kernel. We do not implement these optimizations in the reported benchmarks.

5 Discussion

The usefulness of very short convolutions is somewhat surprising. With typical kernel sizes of only a few tokens, short convolution provides a much narrower interaction pattern than attention or recurrent sequence mixing, yet it consistently improves modern sequence models (Allen-Zhu, 2025; Sieberling et al., 2026; Tian et al., 2026). This suggests that its benefit may not arise only from local mixing in the usual sense. Instead, short convolution may alter both the effective workspace available to each position and the form in which local information is exposed to downstream computation.

We hypothesize that short convolution provides each position with a distributed local workspace. Because its reads are unconditional, nearby states are reliably accessible, so a position can treat them as extensions of its own state: downstream computation becomes a function of the local window rather than of the current residual state alone. No individual state then has to hold everything its consumers require. This resembles how attention typically relieves the single-state context bottleneck, but differs: what attention retrieves must still be written into the consuming state to be used, whereas here the neighbor's state is read where it is used and need not occupy the consuming state.

This relief reflects a general coupling between access and storage in the residual stream: if a computation can access only the current residual state, any information it uses must first be written into that state, where it both occupies finite capacity and is exposed to every downstream consumer. Under a shared and finite state, broadly useful information is the most natural to persist, whereas information useful to only a few computations is less likely to justify a lasting place — and it is exactly this narrowly useful information that reading at the point of use serves: it can be read where it helps without being stored where it mostly would not.

Crucially, this benefit comes not from the operator itself but from where its output goes: because the output feeds computation within the block instead of being written back to the residual stream, information can inform a computation without ever being committed to the consuming state. Any sequence-mixing operation whose output is consumed within the block would share this interpretation, with short convolution a particularly cheap realization; short sliding window attention, for

instance, has been proposed as a drop-in replacement at this placement (Zhang, 2025). Placements vary considerably (Tian et al., 2026), and we intend these hypotheses to apply where this property holds rather than to short convolution as a primitive.

The account so far concerns where information is stored and accessed, but short convolution may also provide a more natural input to local computation. Some useful features are inherently relations among neighboring states, not properties of a single state. Without direct local access, the model must first fold the relevant information into one position and only then compute the relation. Short convolution exposes the local window in an offset-specific form within the block, allowing downstream computation to act on that structure more directly. This may make local relational computations easier to express, independently of any reduction in residual-stream storage pressure.

Even within this scope, these interpretations are unlikely to be complete, and they compete with a more conventional family of explanations: by handling local interactions directly, short convolution may simply relieve other components of that burden — attention need not devote heads to adjacent tokens, feed-forward layers need not reconstruct local features, and compositions that would otherwise require additional depth become available earlier. Beyond what short convolution makes available, its locality may also act as an inductive bias that makes local functions easier to learn. These explanations are not mutually exclusive with the interpretations above, and our experiments do not distinguish which contributes most to the observed gains.

Taken together, these considerations point toward a broader view of short convolution than simply local sequence mixing. Understanding how short convolutions interact with residual representations, relational computation, associative memory, and tokenization remains an interesting direction for future work; we provide a preliminary structural analysis of the operators learned by MoC in Appendix D.

6 Related Work

Short convolution in sequence models. Short convolution has appeared in several modern sequence-modeling architectures. Primer introduced depthwise convolution over attention projections as a simple modification to Transformer blocks (So et al., 2021). More recently, short convolutions have become especially common in efficient recurrent and linear-attention architectures, including Mamba-2, Gated DeltaNet, and KDA (Dao & Gu, 2024; Yang et al., 2025; Team et al., 2025). Canon layers provide a broader architectural use of this primitive, inserting short convolutions at several points within sequence-modeling blocks (Allen-Zhu, 2025). Recent work has also investigated static short convolution within Transformer blocks, performing extensive studies of convolution placement and module design for modern LLMs (Tian et al., 2026). An interesting direction for future work is to investigate MoC across a broader range of insertion points.

Dynamic convolution. Dynamic convolution has been explored in several forms. Dynamic Filter Networks generate input-conditioned filters directly (De Brabandere et al., 2016), while CondConv learns example-dependent convolutional kernels through conditional parameterization (Yang et al., 2019). Dynamic Convolution forms input-dependent kernels as attention-weighted mixtures of learned convolutional bases (Chen et al., 2020); MoC uses a closely related mixture parameterization, but applies it independently at each sequence position to short causal depthwise convolution. Selective Kernel Networks dynamically fuse convolutional branches with different receptive fields (Li et al., 2019). In sequence modeling, Lightweight and Dynamic Convolutions introduced dynamic convolution as an efficient alternative to self-attention (Wu et al., 2019). MoC differs in architectural role: rather than serving as a convolutional backbone or attention replacement, it is used as an adaptive replacement for the short convolution already present inside modern sequence blocks.

Dynamic short convolution. Jet-Nemotron introduced an input-conditioned short convolution within JetBlock, using a kernel generator to produce dynamic convolutional weights for the value representations of a linear-attention block (Gu et al., 2025). This mechanism was developed as one component of a broader architecture-search and distillation framework rather than studied as a general replacement for static short convolution during pretraining. Concurrent work by Sieberling et al. (2026) independently studies dynamic short convolution as a general sequence-modeling

primitive during pretraining. In addition to applying dynamic convolution to query, key, and value representations, they show further gains when extending it across the linear layers of a Transformer. MoC is most closely related to their low-rank dynamic-convolution formulation. The two operators admit closely related algebraic forms but motivate different parameterizations: LRDC can be viewed as generating each dynamic kernel through a low-rank linear hyper-network, whereas MoC is formulated as a dense softmax-routed mixture over explicit convolutional experts.

Hyper-connections and stream expansion. Hyper-connections modify residual information flow by introducing multiple residual streams per token and learning how blocks read from and write back to these streams (Zhu et al., 2025). In this sense, they provide a useful point of comparison for short convolution: both mechanisms can be viewed as expanding the set of representational states available to a position before applying a learned aggregation. The expansion occurs in different dimensions, however. Hyper-connections create multiple persistent lanes within each token, whereas short convolution exposes the neighboring token states in a causal local window. Hyper-connections also define both read and write mixing over their residual lanes. For the post-QKV placement studied in this work, short convolution primarily changes the local readout: each position reads from multiple neighboring states, but the resulting output is returned to the corresponding position. MoC follows this same local-window structure, but makes the readout token-adaptive, making it closer in spirit to dynamic hyper-connections while remaining an operator-level modification rather than a structural residual-topology change.

7 Limitations

Our experiments are limited to model scales up to approximately 460M parameters and training budgets up to 15B tokens. Although the results are consistent across two architectures and multiple settings, it remains unclear whether the improvements persist at substantially larger scales. Our primary comparison is between MoC and static short convolution. We also compare against concurrent low-rank dynamic convolution, where MoC obtains lower language-modeling perplexity in our matched settings, but the margins are smaller than the gains over static convolution. Larger-scale comparisons between alternative dynamic short-convolution parameterizations remain an important direction for future work.

Our optimization analysis is conducted primarily with 2.5B-token runs and under the normalization and optimizer choices used in this work. The observed interaction between initialization scale and convolution learning rate may change under substantially longer training, and may differ between static convolution and MoC. Under the AdamW settings studied here, fan-in initialization requires a substantially larger convolution-specific learning rate than small initialization to achieve comparable performance. This coupling may make the optimization recipe less straightforward to transfer to different architectures or optimization settings.

Finally, we evaluate MoC only for autoregressive language modeling with Transformer and Gated DeltaNet backbones. Its behavior in other sequence domains and under different convolution placements remains to be established.

8 Conclusion

We introduced Mixture of Convolutions (MoC), a token-adaptive generalization of short causal depthwise convolution. MoC replaces the single static kernel with a mixture over learned convolutional bases while preserving the causal, depthwise, and local structure of the original operator. Evaluated in Transformer and Gated DeltaNet models, MoC broadly improves language-modeling perplexity over static short convolution, suggesting that token-adaptive local readout is a simple and effective upgrade to a common sequence-modeling primitive.

AI Use Statement

Generative AI tools were used for manuscript drafting and editing, identifying relevant prior work, and assistance with implementation and optimization of the custom Triton kernels used for MoC. All AI-assisted material was reviewed and verified by the authors.

Reproducibility Statement

Section 4.1 and Appendix A provide the model architectures, training setup, optimization details, initialization schemes, and implementation and hardware settings used in our experiments. Additional ablations and comparisons are provided in Appendices B–D.

References

Zeyuan Allen-Zhu. Physics of language models: Part 4.1, architecture design and the magic of canon layers. In *Proceedings of the 39th Conference on Neural Information Processing Systems*, NeurIPS '25, 2025. Full version available at https://ssrn.com/abstract=5240330.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, Los Alamitos, CA, USA, July 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.195. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.195.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 667–675, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, December 2023. URL https://doi.org/10.5281/zenodo.10256836.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. Jet-nemotron: Efficient language model with post neural architecture search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=WZQXaTNYEB.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 25333–25369. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kosson24a.html.

Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 510–519, 2019. doi: 10.1109/CVPR.2019.00060.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In Mirella Lapata, Phil Blunsom, and Alexander Koller (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025/.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1b7whO4SfY.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Oliver Sieberling, Bharat Runwal, Rameswar Panda, and Yoon Kim. Dynamic short convolutions improve transformers. *arXiv preprint arXiv:2606.03825*, 2026.

David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Searching for efficient transformers for language modeling. *Advances in neural information processing systems*, 34:6010–6022, 2021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Yuchuan Tian, Yingte Shu, Wei He, Shuo Zhang, Tianchen Zhao, Chao Xu, Xinghao Chen, Yunhe Wang, Hanting Chen, and Yu Wang. Convolution for large language models. *arXiv preprint arXiv:2607.18413*, 2026.

Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

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

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkVhlh09tX.

Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. *CondConv: conditionally parameterized convolutions for efficient inference*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Songlin Yang and Yu Zhang. FLA: A Triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Yifan Zhang. Rethinking swa. *yifanzhang-pro.github.io*, December 2025. URL https://github.com/yifanzhang-pro/Rethinking-SWA.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9FqARW7dwB.

A Additional Experimental Details

A.1 Model Architecture and Optimization

Table 7 summarizes the model architectures and optimization settings used in our experiments unless stated otherwise.

Table 7: Additional architecture and optimization details. Here, d denotes the model dimension. Settings shared across model sizes are listed once.

Setting	Value
Model architecture	
110M configuration	$d = 512$, 12 layers
190M configuration	$d = 768$, 12 layers
460M configuration	$d = 1024$, 24 layers
Context length	2048
Block structure	Llama-style pre-normalization (Grattafiori et al., 2024)
Feed-forward network	SwiGLU with intermediate dimension $3d$ (Shazeer, 2020)
Transformer attention heads	Head dimension 64; number of heads $d/64$
Transformer KV heads	Equal to the number of query heads unless otherwise stated
Gated DeltaNet heads	4 heads; query/key head dimension $d/8$; value head dimension $d/4$
Normalization	RMS normalization (Zhang & Sennrich, 2019)
Embedding normalization	RMS normalization immediately after embedding lookup
Retrieved value normalization	Per-head RMS normalization, applied to the sequence-mixing output
Gated attention	SDPA per-head sigmoid formulation of Qiu et al. (2025)
Positional encoding	RoPE (Su et al., 2024) with base 10,000
Tokenizer	Llama 3 tokenizer with vocabulary size 128,000 (Grattafiori et al., 2024)
Embeddings	Tied input and output embeddings (Press & Wolf, 2017)
Linear-layer biases	Disabled, except for the LRDC bias
Short-convolution configuration	
Kernel size	$S = 4$
Placement	Applied to the projected query, key, and value representations
Transformer query/key ordering	Convolution, query-key normalization, then RoPE
Initialization	
Linear weights	Normal initialization with standard deviation 0.02
Static convolution	Uniform initialization with standard deviation 0.002
MoC bases	Uniform initialization with standard deviation 0.002
MoC router	Zero-initialized
LRDC	Initialized following Sieberling et al. (2026)
Optimization	
AdamW (Loshchilov & Hutter, 2019) parameters	Embeddings, static conv, MoC bases, LRDC U and bias, and 1D parameters
AdamW learning rate	10^{-3}
AdamW coefficients	$(\beta_1, \beta_2) = (0.9, 0.95)$
NorMuon (Li et al., 2025) parameters	2D parameters excluding those mentioned in AdamW
NorMuon learning rate	10^{-2}
NorMuon settings	Momentum 0.95, $\beta_2 = 0.95$, and Nesterov acceleration
Weight decay	0.1; one-dimensional parameters are excluded
Global batch size	524,288 tokens
Gradient clipping	Global norm 1.0
Learning-rate schedule	2% linear warmup followed by cosine decay to zero
Implementation and hardware	
Implementation	PyTorch with `torch.compile`
Training precision	bfloat16
Hardware	1x NVIDIA RTX 5090 GPU

B Comparison with Low-Rank Dynamic Convolution

MoC and LRDC (Sieberling et al., 2026) differ primarily in their routing parameterization and associated architectural choices. Ignoring these differences, the two methods perform closely related adaptive short-convolution operations. LRDC combines its learned components through unconstrained routing weights, whereas MoC uses positive basis coefficients and, in its standard form, normalizes them to produce a convex combination of learned kernels.

Although this architectural difference is small, it changes the surrounding design choices that lead to good performance. In our experiments, the static kernel component and explicit residual connection used by LRDC consistently degraded MoC performance. Conversely, the initialization scheme proposed in Section 4.3 transfers only marginally to LRDC, since all of its dynamic low-rank projections are already initialized to zero and only the static kernel component, implemented as a bias on the generated kernel, receives a nontrivial initialization.

We therefore additionally evaluate reducing the initialization of the static kernel component from the original fan-in scale (0.29) to 0.002.

Table 8: Effect of reducing the initialization of the LRDC static kernel component. All other parameters retain the original LRDC initialization. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Static kernel init. std.	F-Edu ppl↓
0.29	24.20
0.002	24.15

The improvement is modest, consistent with LRDC already employing an effectively small initialization for its dynamic components. We retain the original LRDC initialization from the published implementation in the main experiments. For the matched-seed comparison below, however, we use the slightly improved static-kernel initialization to better align the optimization settings of the two methods.

Table 9: Three-seed comparison between LRDC and MoC at 110M parameters and 2.5B training tokens.

Method	Seed 1	Seed 2	Seed 3
LRDC	24.15	24.11	24.14
MoC	23.79	23.87	23.79

Using this slightly improved LRDC configuration, we compare LRDC and MoC over three matched random seeds at 110M parameters and 2.5B training tokens. Across all three seeds, every MoC run achieves lower validation perplexity than every LRDC run.

Because MoC and LRDC differ not only in the routing nonlinearity but also in surrounding architectural choices, we next isolate the effect of the MoC routing activation in a fixed MoC-style architecture. All MoC-style variants omit the static kernel bias and explicit residual connection, initialize the router with standard deviation 0.02, and initialize the basis kernels from $\mathcal{U}(-0.002\sqrt{3}, 0.002\sqrt{3})$. For this ablation, we do not renormalize the routing weights to sum to one after applying the activation; this normalization is mostly redundant here because the convolution outputs are subsequently normalized. Table 10 compares several routing activations under this fixed architecture.

Within the fixed MoC-style architecture, positive routing weights appear important. The linear variant performs worse than LRDC, indicating that the two methods are not made equivalent simply by removing the softmax; the surrounding architectural choices also matter. ELU, whose outputs can be negative, performs worse than the linear variant, whereas ELU$+1$, softplus, and exponential routing perform comparably. We observe some differences in training dynamics: ELU$+1$ and softplus perform slightly better earlier in training, while exponential routing progressively closes this gap. We additionally test a shifted-softplus router, adding a single learned scalar shift to all router preactivations. The learned shift becomes progressively more negative during training, moving more

Table 10: Effect of routing parameterization at 190M parameters and 5B training tokens. LRDC uses its original architectural choices, including the static kernel bias and explicit residual connection. The remaining rows use a fixed MoC-style architecture without a static kernel bias or explicit residual connection, with router initialization standard deviation 0.02 and basis-kernel initialization standard deviation 0.002. We report FineWeb-Edu validation perplexity.

Architecture	Routing activation	F-Edu ppl↓
LRDC	Linear, with LRDC design choices	18.87
MoC-style	Linear	19.17
MoC-style	ELU	19.23
MoC-style	ELU+1	18.70
MoC-style	Softplus	18.70
MoC-style	Exp.	18.72

router inputs into the negative, exponential-like region of the softplus rather than its positive, approximately linear region. This suggests that positivity is the clearest property identified by this ablation, while the exponential-like regime may also be useful among positive routing functions.

We then isolate the role of routing-weight normalization itself by removing the post-convolution output normalizations, so that the softmax denominator is no longer absorbed by subsequent normalization. We compare softmax routing with unnormalized exponential routing in the same MoC-style architecture. Because the output normalizations are removed, both variants use fan-in-scale basis initialization to maintain the initial convolution output scale. For $k = 16$, softmax routing uses basis initialization standard deviation 1.15, while unnormalized exponential routing uses standard deviation 0.07. The unnormalized exponential initialization is therefore $16\times$ smaller, since it does not divide the routing weights by their sum. We reduce the convolution learning rate by the same factor, using 0.16 for softmax routing and 0.01 for unnormalized exponential routing, to maintain the initial optimizer update relative to the parameter scale.

Table 11: Effect of routing-weight normalization when post-convolution output normalization is removed. Both variants use fan-in-scale basis initialization to maintain the initial convolution output scale. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Routing	Basis init. std.	Conv. LR	F-Edu ppl↓
Exp.	0.07	0.01	24.31
Softmax	1.15	0.16	24.21

Without post-convolution output normalization, unnormalized exponential routing is no longer equivalent to softmax routing. Softmax routing performs better, indicating that routing-weight normalization provides an additional benefit when it is not made redundant by subsequent output normalization.

C Additional Optimization Results

C.1 Learning Rate at Fan-In Initialization

The matched convolution learning rate in Section 4.3 is chosen to preserve the initial update-to-parameter ratio of the small-initialization configuration; it is not selected by tuning validation performance. Because the resulting learning rates are much larger than those typically used for these parameters, we conduct a coarse sweep over the MoC convolution learning rate while holding the fan-in initialization fixed.

Performance improves substantially as the convolution learning rate increases beyond the baseline value, then remains nearly flat from 0.58 through 4.0. This indicates that the ratio-matched learning rate is not a precise optimum; rather, it moves fan-in initialization into a broad well-performing

Table 12: Coarse sweep over the MoC convolution learning rate at fan-in initialization. All configurations use the MoC fan-in initialization with standard deviation 1.15. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Conv. LR	F-Edu ppl↓
0.001	25.32
0.01	24.48
0.1	23.94
0.58	23.83
1.0	23.80
2.0	23.81
4.0	23.82

high-learning-rate regime. This robustness to very large convolution learning rates likely depends on the normalization choices used in our models. In particular, without query-key normalization, the matched learning rate 0.58 produces loss spikes in preliminary runs, whereas the smaller learning rate 0.16 used in Table 11 remains stable. Because weight decay can interact with the effective update size during training, the precise location of this regime may also depend on the weight-decay setting (Van Laarhoven, 2017; Kosson et al., 2024).

C.2 Retrieved Value Normalization

Normalization maintains representation scale when short-convolution parameters are initialized at the small scale used in our main experiments. Query-key normalization is a more standard component of the Transformer architecture, whereas retrieved value normalization is not. We therefore test whether retrieved value normalization remains necessary when MoC instead uses fan-in initialization and a matched convolution learning rate.

Table 13: Effect of retrieved value normalization under different MoC optimization settings. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Retrieved value norm.	Init. std.	Conv. LR	F-Edu ppl↓
Yes	0.002	0.001	23.79
Yes	1.15	0.58	23.83
No	1.15	0.58	23.80

With fan-in initialization and the matched convolution learning rate, removing retrieved value normalization has a negligible effect on validation perplexity. Retrieved value normalization is therefore not required for MoC under this optimization configuration.

C.3 Conservative Optimization Recipe

To check that the dependence on initialization scale is not specific to the optimization recipe used in our main experiments or to the Transformer backbone, we repeat the comparison under a more conservative setting: AdamW for all parameters at a learning rate of 3×10^{-4}, in both Transformer and Gated DeltaNet models.

Table 14: Effect of static short-convolution initialization scale under a uniform AdamW recipe. All parameters are optimized with AdamW at a learning rate of 3×10^{-4}. We report FineWeb-Edu validation perplexity for 110M-parameter models trained on 2.5B tokens.

Init. std.	Transformer	Gated DeltaNet
0.29	38.20	38.29
0.002	35.13	36.95

Smaller initialization improves static short convolution in both architectures under this recipe, with a larger gain for Transformer. Notably, the training curves differ as well: in Transformer, small initialization takes the lead over fan-in initialization almost immediately, whereas in Gated DeltaNet it remains behind for several hundred optimizer steps before the two cross over. We investigate this difference briefly in Appendix C.4.

C.4 Block Structure and Initialization Scale

We attempt to isolate which differences between the Transformer and Gated DeltaNet blocks are responsible for the smaller benefit of small initialization in Gated DeltaNet, beginning with the post-convolution nonlinearity. Gated DeltaNet applies a SiLU nonlinearity to the projected query, key, and value representations after the short convolution, whereas our Transformer blocks apply no nonlinearity at this position. We compare static short convolution with and without this nonlinearity under fan-in and small initialization.

Table 15: Interaction between static short-convolution initialization and the post-convolution nonlinearity. All configurations use a convolution learning rate of 10^{-3}. We report FineWeb-Edu validation perplexity for 190M-parameter Gated DeltaNet models trained on 5B tokens.

Init. std.	With SiLU	Without SiLU
0.29	19.56	19.75
0.002	19.45	19.48

Small initialization improves performance both with and without the post-convolution nonlinearity, but the relative benefit increases when it is removed, indicating that the nonlinearity is partly responsible for the smaller gain. It likely does not account for the difference entirely: fan-in initialization still leads for a few hundred steps, although the crossover occurs earlier.

Viewed along the other axis, removing the nonlinearity degrades performance under fan-in initialization, consistent with its use in Gated DeltaNet and related architectures (Dao & Gu, 2024; Team et al., 2025), which initialize short convolutions at the fan-in scale. Under small initialization, however, removing it performs comparably to retaining it. The nonlinearity may therefore compensate for large initialization, addressing much the same deficiency as small initialization, or the two may provide distinct benefits that do not combine. We do not have a mechanistic account that distinguishes these.

Precisely isolating the remaining difference is left to future work. We remove the nonlinearity from the query, key, and value paths uniformly and do not determine whether one path is responsible, nor do we vary the initialization scale separately across the three paths, which would provide another way to adjust the relative update magnitude within the block. We also do not vary the SiLU output gate, which may make small initialization too aggressive in terms of relative updates; testing this without the output gate, or with a more constrained gate such as a per-head sigmoid (Qiu et al., 2025) or the low-rank sigmoid gate used in KDA (Team et al., 2025), would help separate the two. Other differences between the backbones, including the block configuration and the sequence-mixing operation itself, may also contribute.

C.5 Fixed Uniform Routing

To separate the effect of token-adaptive routing from the effect of using multiple convolutional bases, we evaluate a fixed-routing MoC variant. This model uses the same basis-bank parameterization as MoC, but removes the router and instead averages the bases with fixed uniform weights at every token. The resulting operator is therefore equivalent to a single learned static short-convolution kernel, up to its overparameterized basis representation.

Fixed uniform routing performs substantially closer to static short convolution than to MoC. This indicates that the improvement from MoC is not explained simply by the presence of multiple learned convolutional bases; the token-dependent routing contributes materially to the gain.

Table 16: Effect of removing token adaptivity from MoC. Fixed uniform routing uses the same basis-bank parameterization as MoC, but averages the bases with fixed uniform weights at every token. We report FineWeb-Edu validation perplexity for 110M-parameter Transformer models trained on 2.5B tokens.

Method	F-Edu ppl↓
Static conv.	24.21
Fixed uniform routing	24.15
MoC	23.79

D Learned Operator Analysis

We provide a coarse structural summary of the convolutional bases learned by MoC. For each channelwise length-four causal kernel w, we assign each entry a symbol in $\{-, 0, +\}$. We display patterns in temporal order from oldest to newest context position, so the rightmost symbol corresponds to the weight applied to the current token. Entry w_s is treated as zero when

$$|w_s| \le \max\left(10^{-4}, 0.1 \max_j |w_j|\right).$$

Patterns that differ only by a global sign inversion are merged, and one canonical representative is displayed for each equivalence class. Thus, ++++, for example, represents both all-positive and all-negative kernels. Frequencies are pooled across all layers, bases, and channels of the 460M Transformer model.

Because MoC forms token-specific kernels as convex combinations of explicit learned bases, these summaries provide a coarse view of the local operator structures available to the router. However, they do not incorporate weight magnitudes or token-dependent routing probabilities and therefore do not fully describe the effective kernels used at individual tokens.

Table 17: Distribution of discretized sign structures among learned MoC bases. Globally sign-inverted patterns are merged. Frequencies are computed separately for query, key, and value convolutions over all channelwise kernels across layers and bases. Rows are ordered by mean frequency across the three projections.

Pattern	Query	Key	Value
++++	16.30%	16.57%	8.82%
+++−	9.26%	8.75%	9.03%
++−−	6.49%	7.01%	7.40%
++−+	5.68%	6.38%	8.77%
+−−−	4.59%	4.99%	3.28%
+−−+	3.86%	4.54%	4.04%
0++−	3.87%	3.37%	5.00%
00+−	3.03%	2.11%	6.97%
0+++	4.71%	4.49%	2.82%
+−++	3.15%	4.05%	3.17%
0+−+	2.00%	2.33%	5.55%
+−+−	2.74%	3.41%	3.38%
00++	3.55%	2.47%	2.53%
0+−−	2.47%	2.72%	3.05%
+0−−	2.74%	2.25%	1.97%
+0++	2.76%	2.58%	1.60%
+0−+	2.02%	1.99%	2.80%
+++0	1.74%	2.73%	1.87%
+0+−	1.96%	1.77%	2.49%
++0+	2.50%	1.76%	1.26%

Continued on next page

Table 17 – continued from previous page

Pattern	Query	Key	Value
++0−	2.23%	1.52%	1.39%
++−0	0.95%	1.66%	2.08%
000+	2.56%	0.58%	0.62%
00+0	0.35%	0.74%	1.64%
+−−0	0.65%	1.14%	0.87%
0++0	0.58%	1.14%	0.93%
0+−0	0.33%	0.70%	1.41%
0+0−	1.19%	0.59%	0.58%
+−0+	0.80%	0.84%	0.69%
+−+0	0.49%	0.83%	0.84%
+−0−	0.79%	0.74%	0.56%
0+0+	0.93%	0.57%	0.46%
+00−	0.98%	0.37%	0.37%
+00+	0.95%	0.41%	0.28%
+0−0	0.24%	0.50%	0.64%
+0+0	0.23%	0.45%	0.45%
++00	0.21%	0.43%	0.18%
+−00	0.08%	0.21%	0.14%
0+00	0.03%	0.29%	0.07%
+000	0.02%	0.02%	0.01%

Query and key bases exhibit closely related distributions, with the same-sign pattern forming the largest individual class. Value bases are less concentrated on this structure and assign substantially more mass to several mixed-sign and partially sparse patterns. More broadly, every possible nonzero relative sign structure appears with nonzero frequency, indicating that the learned bases span a broad set of local aggregation, differencing, and selective-tap structures rather than collapsing to a small number of templates.

To obtain a preliminary view of token-dependent routing behavior, we cluster routing-probability vectors using k-means and qualitatively inspect both the tokens represented by each cluster and the sign structures of experts receiving higher routing probability within that cluster. Although we observe many recurring routing patterns, these are not always consistent in either the token types they capture or the effective adaptive operators they produce. One notable exception involves fragmented BPE tokens: clusters containing trailing subword fragments are frequently associated with structures such as ++−+, ++−−, and 00+−. We do not establish that these structures are specific to fragmented tokens, and they may also serve other roles. We therefore report this only as a preliminary qualitative indication that some learned local operators respond systematically to tokenization structure.