MambaExtend: A Training-Free Approach to Improve Long Context Extension of Mamba

Seyedarmin Azizi[1], Souvik Kundu[2], Mohammad Erfan Sadeghi[1], Massoud Pedram[1]

[1]University of Southern California, USA [2]Intel Labs, San Diego, USA

Motivation

- Transformer-based LLMs struggle with long contexts due to quadratic complexity.

- Mamba (SSM-based) improves efficiency, but still fails on long sequences accuracy-wise.
- Key limitation: out-of-distribution discretization steps (Δt) at inference time.

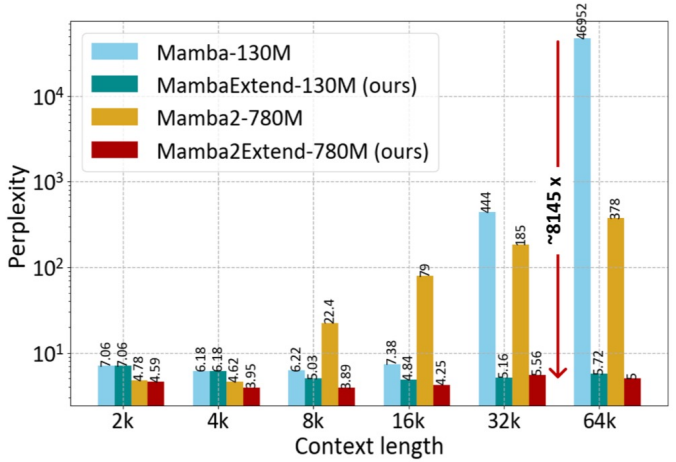

Figure 1: Long-context understanding on Pile. Compared to the pre-trained alternatives, MambaExtend provides up to $\sim\mathbf{8145}\times$ improvement in perplexity score, via a **training-free** calibration.

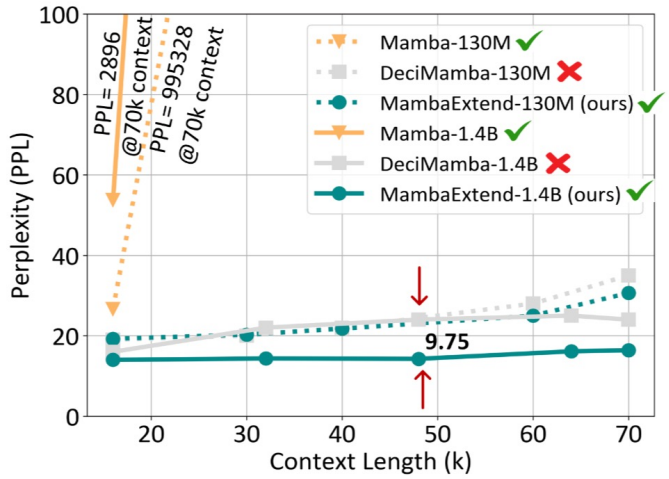

Figure 4: Perplexity comparison on PG-19. The ✓ and ✗ identify the fine-tuning requirements to be false and true, respectively.

SSM Summary

$$h_t = \bar{A}h_{t-1} + \bar{B}z_t, o_t = Ch_t$$

$$\bar{A}_t = \texttt{exp}(\Delta_t A), \bar{B}_t = \Delta_t B_t \text{ where } \Delta_t = \texttt{SFT}(\Delta_{t_{proj}}(z_t)), B_t = W_B(z_t), C_t = (W_C(z_t))^T$$

S6 layer input-output behavior:

$$O = \alpha Z \text{ with } \alpha_{i,j} = C_i \left(\prod_{k=j+1}^{i} \bar{A}_k \right) \bar{B}_j \longrightarrow \alpha_{P,1} = C_P \prod_{k=2}^{P} \bar{A}_k \bar{B}_1 = C_P \texttt{exp}(A \sum_{k=2}^{P} \Delta_k) \bar{B}_1$$

$$\begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ o_P \end{pmatrix} = \begin{pmatrix} C_1\bar{B}_1 & 0 & \cdots & 0 \\ C_2\bar{A}_2\bar{B}_1 & C_2\bar{B}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_P \prod_{k=2}^{P} \bar{A}_k\bar{B}_1 & C_P \prod_{k=3}^{P} \bar{A}_k\bar{B}_2 & \cdots & C_P\bar{B}_P \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_P \end{pmatrix}$$

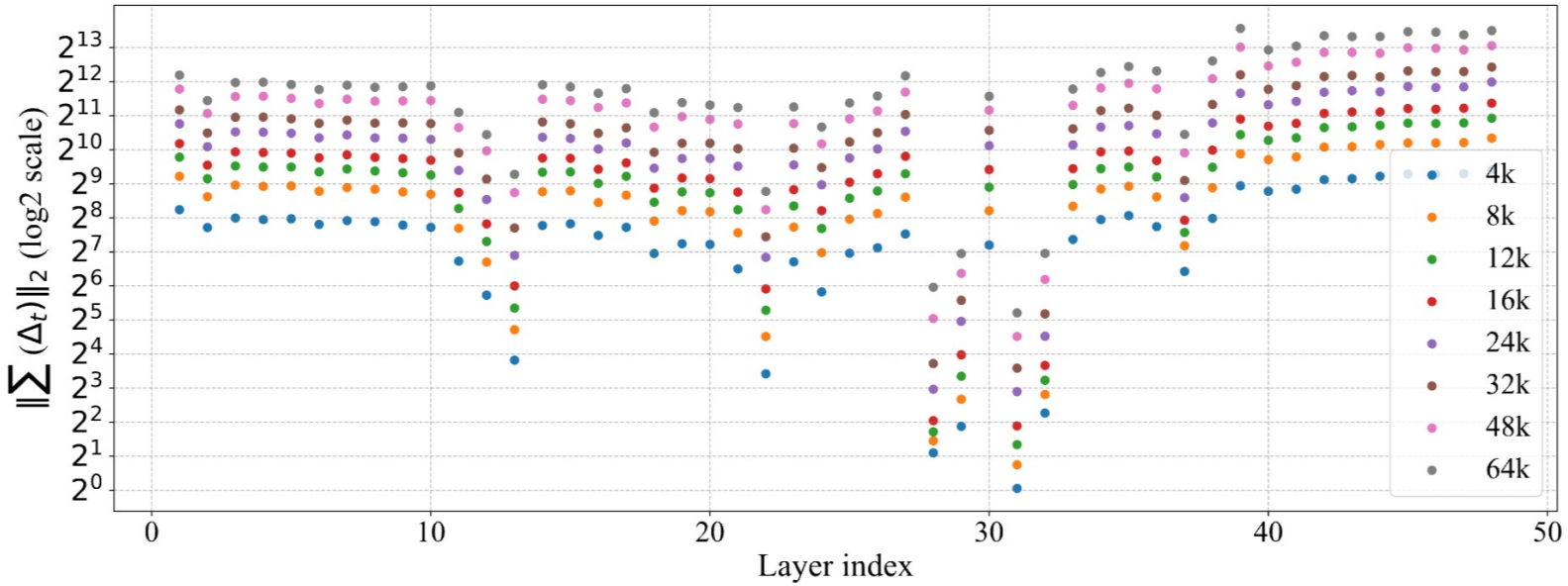

Figure 2: Layer-wise behavior of $\sum(\Delta_t)$ for different context length during test-time. We used the Pile dataset on Mamba 1.4B for the evaluation.

MambaExtend: Our Contributions

- Training-free framework to calibrate Mamba for long contexts.
- Scales discretization steps (Δt) per layer using learnable factors, resembling position interpolation in transformers.

- Uses both gradient-based and zeroth-order optimization.
- Methodology overview:
  - For each Mamba layer: learn a scaling factor for Δt.
  - Keeps original model weights frozen.
  - Optimizes only ∼L parameters → very efficient.
  - Supports inference up to 64k tokens (32× extension).

Algorithm 1 MambaExtend Algorithm

Input: An L-layer Mamba model parameterized by $\mathcal{M}$, set of calibration samples $\mathcal{C}$, calibration function `CF`
Output: Scaling factors $\mathbf{S} = [s_1, s_2, ..., s_L]$, where $s_i \in \mathbb{R}^m$
for $i \leq L$ **do**
    $s_i \leftarrow \texttt{init}(U(0,1))$
end for
`freeze`$(\mathcal{M})$
$\mathbf{S} \leftarrow \texttt{CF}(\mathbf{S}, \mathcal{C}, \mathcal{M})$
return S

Calibration functions

Algorithm 2 $\texttt{CF}_{BP}$ Algorithm

Input: An L-layer Mamba model parameterized by frozen weights $\mathcal{M}$, set of calibration samples $\mathcal{C}$, the initialized scaling factors $\mathbf{S}$
Input: Learning rate η, number of iterations K
Output: Learned Scaling factors $\mathbf{S} = [s_1, s_2, ..., s_L]$, where $s_i \in \mathbb{R}_+^m$
`optimizer` = Adam($\mathbf{S}, \eta$)
for $k \leq K$ **do**
    $\mathcal{L}$ = `Evaluate`($\mathcal{M}_{\Delta_t \times \mathbf{S}}, \mathcal{C}$)
    $\mathcal{L}$.`backward()`
    `optimizer.step()`
    $\mathbf{S} \leftarrow \mathbf{S}$.`clamp`(min $= 0.001$) # make sure scaling factors remain positive
end for
return $\mathbf{S}$

Algorithm 3 $\texttt{CF}_{ZO}$ Algorithm

Input: An L-layer Mamba model parameterized by $\mathcal{M}$, set of calibration samples $\mathcal{C}$, the initialized scaling factors $\mathbf{S}$
Output: Learned scaling factors $\mathbf{S} = [s_1, s_2, ..., s_L]$, where $s_i \in \mathbb{R}_+^m$
Specify learning rate η, perturbation magnitude c, number of iterations K
for k $\leq$ K **do**
    $\delta \in \mathbb{R}^{L \times m} \sim$ `Rademacher()`
    $\mathbf{S}^+ = \mathbf{S} + c \times \delta, \quad \mathbf{S}^- = \mathbf{S} - c \times \delta$
    $\mathcal{L}^+ =$ `Evaluate`$(\mathcal{M}_{\Delta_t \times \mathbf{S}^+}, \mathcal{C}), \quad \mathcal{L}^- =$ `Evaluate`$(\mathcal{M}_{\Delta_t \times \mathbf{S}^-}, \mathcal{C})$
    $\hat{\nabla}_{\mathbf{S}} = (\mathcal{L}^+ - \mathcal{L}^-)/(2c\delta)$
    $\mathbf{S} \leftarrow \mathbf{S} - \eta \hat{\nabla}_{\mathbf{S}}$
    $\mathbf{S} \leftarrow \mathbf{S}$.`clamp`(min $= 0.001$) # make sure scaling factors remain positive
end for
return $\mathbf{S}$

- Experimental Results

Table 1: Perplexity for Mamba models over different evaluation context lengths on Pile dataset.

	Mamba-130M						Mamba-1.4B						Mamba2-780M					
Context Length	**2k**	**4k**	**8k**	**16k**	**32k**	**64k**	**2k**	**4k**	**8k**	**16k**	**32k**	**64k**	**2k**	**4k**	**8k**	**16k**	**32k**	**64k**
Pre-trained Model	7.06	6.18	6.22	7.38	444	46592	4.34	3.78	4.19	14.4	260	6304	4.78	4.62	22.4	79	185	378
MambaExtend	7.06	6.18	**5.03**	**4.84**	**5.16**	**5.72**	**4.31**	3.78	**3.48**	**3.62**	**4.81**	**6.93**	**4.59**	**3.95**	**3.89**	**4.25**	**5.56**	**5.00**

Table 2: Mamba vs MambaExtend performance on representative LongBench tasks.

Model	**Qasper**	**HotpotQA**	**2WikiMultihopQA**	**TREC**	**TriviaQA**	**LCC**	**RepoBench-P**	**Average**
Mamba-1.4B	7.0	11.00	9.75	29.00	1.67	20.12	11.67	12.88
MambaExtend-1.4B	**16.67**	**14.29**	**13.82**	**35.0**	**7.67**	**26.12**	**18.84**	**18.91**
Mamba2-780M	7.50	6.06	9.48	17.0	0.1	22.1	14.01	10.89
MambaExtend2-780M	**7.96**	**10.95**	**18.33**	**28.00**	**6.83**	**28.27**	**17.71**	**16.86**

Figure 5: Passkey retrieval performance after fine-tuning (FT) (for Mamba and DeciMamba) or calibrating (for MambaExtend) on samples of 4k context length.

- Efficiency of MambaExtend

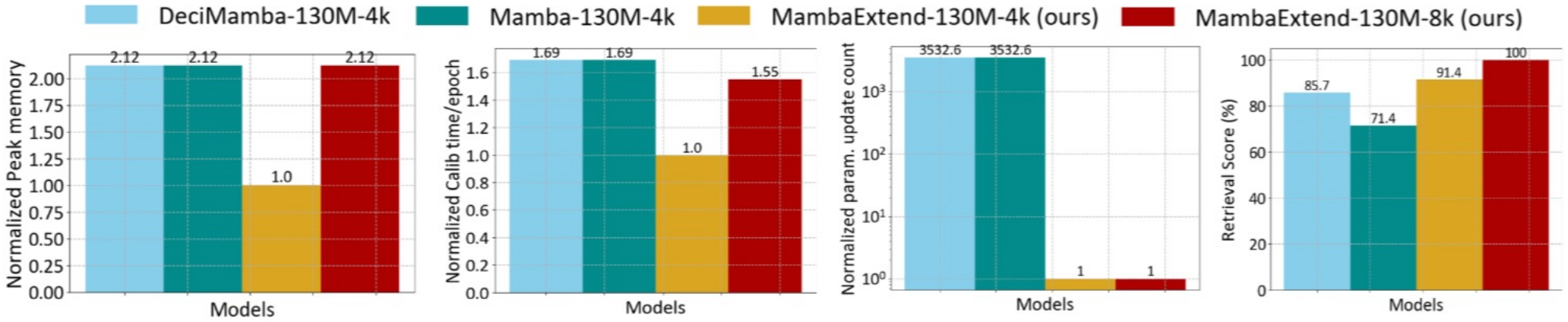

Figure 6: Comparison of normalized {*peak memory, calibration time, and number of parameter updates*} between Mamba, DeciMamba, and MambaExtend for passkey retrieval task. We use Mamba-130M model and for each method, we train for *one* epoch either with 4k or with 8k context length. For these three measurement types, we normalize each value by the corresponding value of MambaExtend-130M-4k.

Thank You!