Slender-Mamba: Fully Quantized Mamba in 1.58 Bits From Head to Toe

Zhenxuan Yu[*]
Rikkyo University

Takeshi Kojima[†]
The University of Tokyo,

Yutaka Matsuo[†]
The University of Tokyo,

Yusuke Iwasawa[†]
The University of Tokyo,

Abstract

Large language models (LLMs) have achieved significant performance improvements in natural language processing (NLP) domain. However, these models often require large computational resources for training and inference. Recently, Mamba, a language model architecture based on State-Space Models (SSMs), has achieved comparable performance to Transformer models while significantly reducing costs by compressing context windows during inference. We focused on the potential of the lightweight Mamba architecture by applying BitNet quantization method to the model architecture. In addition, while prior BitNet methods generally quantized only linear layers in the main body, we extensively quantized the embedding and projection layers considering their significant proportion of model parameters. In our experiments, we applied ternary quantization to the Mamba-2 (170M) architecture and pre-trained the model with 150 B tokens from scratch. Our method achieves approximately 90.0% reduction in the bits used by all parameters, achieving a significant improvement compared with a 48.4% reduction by the conventional BitNet quantization method. In addition, our method experienced minimal performance degradation in both the pre-training perplexity and downstream tasks. These findings demonstrate the potential of incorporating lightweight language models into edge devices, which will become more demanding in the future.[1]

1 Introduction

Large language models (LLMs) have become increasingly prevalent in various aspects of human life, demonstrating impressive capabilities for a wide range of natural language processing tasks (Achiam et al., 2023). LLMs have acquired these capabilities by expanding their model size following scaling laws (Brown et al., 2020).

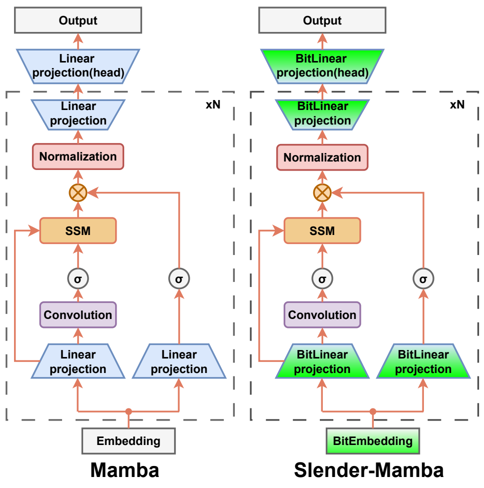

Figure 1: (left) Model structure of Mamba-2 (Dao and Gu, 2024). (right) Model structure of Slender-Mamba, where we replace linear, embedding, and projection layers of Mamba-2 with BitLinear and BitEmbedding.

However, the substantial scale and high energy consumption of current mainstream LLMs present significant environmental challenges and limitations for deployment on edge devices (Strubell et al., 2020). There is a growing demand to develop compact and efficient LLMs, particularly those suitable for edge device applications, expecting small computational costs, high latency speeds, and offline operations simultaneously.

Recent advancements in language model architectures, such as Mamba based on State-Space Models (SSMs) (Gu and Dao, 2023), have gained attention as computationally cost-effective methods at the time of inference. These models have been shown to achieve comparable performance to the Transformer (Vaswani, 2017) while significantly reducing costs by compressing the context length during inference. This approach offers potential

[*]E-mail: `24wr006m@rikkyo.ac.jp`

[†]Email: `{t.kojima, matsuo, iwasawa}@weblab.t.u-tokyo.ac.jp`

[1]Code is available at `https://github.com/YU-ZHENXUAN-ucllm/Slender-Mamba`

advantages in resource-constrained environments.

BitNet (Wang et al., 2023; Ma et al., 2024b) is a lightweight model architecture based on parameter quantization. BitNet b1.58 (Ma et al., 2024b) reduces computational complexity by limiting weights to ternary values {-1, 0, 1}, significantly decreasing memory requirements and bandwidth consumption, while maintaining a performance comparable to that of full-precision models. However, because language models must employ high-precision probabilities for sampling, this technology has only been implemented in the linear layers of the attention block in the Transformer, leaving the embedding and head layers untouched. With the growing demand for multilingual (Qin et al., 2024) and multimodal (Chu et al., 2024) technologies, the parameter size (i.e., vocabulary size) of the embedding and head layers is expanding. In addition, under the condition that vocabulary size is fixed, the parameters of these layers will account for a larger percentage of the total if we try to reduce the overall parameter size in order to develop lightweight model. Therefore, there is an increasing need to quantize these layers.

This study aims to address a new challenge: ***How can we develop models that are more lightweight than Mamba?*** We propose Slender-Mamba, a highly parameter-efficient language model that further advances Mamba's lightweight design by applying BitNet method to the model. While prior BitNet methods quantized only linear layers in the main body, we extensively quantized the embedding and projection layers considering their significant proportion of model parameters. In our experiments, we applied ternary quantization to the Mamba-2 (170M) architecture and pre-trained the model with 150B tokens from scratch. Our method achieves approximately 90.0% reduction in the bits used by all parameters, achieving a significant improvement compared with a 48.4% reduction by the conventional BitNet quantization method. In addition, our method experienced minimal performance degradation in both the pre-training perplexity and downstream tasks.

2 Related Work

Mamba Mamba is one of the innovations of language models based on structured state space models (SSMs) (Gu et al., 2020). The model has been shown to achieve comparable performance with a Transformer (Vaswani, 2017) while significantly reducing costs by compressing the context length during inference. Mamba-1 (Gu and Dao, 2023) combines the H3 block (Fu et al., 2022) with the ubiquitous MLP block of modern neural networks by interleaving them, thereby resulting in a more powerful architecture. Mamba-2 (Dao and Gu, 2024) is constructed using the original Mamba architecture. Similar to its predecessor, Mamba-2 offers a promising alternative to self-attention layers and Transformer in sequence modeling tasks. The model can be computed efficiently during training using hardware-aware algorithms by leveraging modern accelerators, such as GPUs. Mamba-2 introduces a theoretical framework called "structured state space duality" (SSD) that connects SSMs, structured matrices, and variants of attention.

Lightweighting Technologies The existing quantization methods for language models are mostly Post-training quantization (PTQ), such as Absmax (Dettmers et al., 2022a), SmoothQuant (Xiao et al., 2023), and GPTQ (Frantar et al., 2022), among others. These methods are simple and easy to apply because they do not require any changes in the training process or model retraining. However, Reducing precision can lead to greater accuracy loss as the model isn't optimized for quantized representation during training.

However, quantization-aware training (QAT), as exemplified by BitNet, reduces computational complexity by employing 1bit or 1.58-bit weights during the pre-training stage, significantly lowering memory requirements, bandwidth consumption, and energy usage while maintaining a performance comparable to that of full-precision models. Although BitNet has proven to be a promising approach for lightweight Transformer models, its effectiveness has not been validated using Mamba models. Furthermore, prior studies did not conduct quantization experiments on the embedding and head layers. In this study, we pioneered the exploration of quantization techniques for these components, thereby providing a novel contribution to this field. Appendix A further describes light weighting methods other than quantization, such as parameter sharing (Takase and Kiyono, 2021).

3 Proposed Method

3.1 Overall architecture

To achieve full BitNet quantization for the Mamba architecture, we first replaced the linear layers in the main block with BitLinear layers (Wang et al., 2023; Ma et al., 2024b; Zhu et al., 2024), which is a conventional approach to BitNet for Transformer. Second, our method quantizes the embedding and head layers using the BitLinear and BitEmbedding layers. BitEmbedding is our proposed method for quantizing embedding layers in language models, and it will be described in later subsection. We employed a ternary quantization method (BitNet b1.58) instead of binary quantization (1-bit BitNet) because it retains the benefits of the original 1-bit BitNet: requiring almost no multiplication operations for matrix multiplication, while significantly improving the performance of the 1-bit LLMs. An overview of the proposed method is presented in Figure 1.

3.2 BitLinear

BitLinear is an existing quantization approach for a linear layer, originally proposed for Transformer architectures (Ma et al., 2024b). This approach is technically applicable to the linear layers in the Mamba architecture. This subsection describes the formulation of BitLinear as preliminary information. First, the original linear-layer weight matrix $\mathbf{W} \in \mathbb{R}^{D_{in} \times D_{out}}$ is quantized into ternary weights $\widetilde{\mathbf{W}}$ within $\{-1, 0, 1\}$ using the following formula:

$$\widetilde{\mathbf{W}} = \text{clip}\left(\text{Round}\left(\frac{\mathbf{W}}{\beta}\right), -1, 1\right), \quad (1)$$

where the clip(x, a, b) function ensures that the input values x are constrained within the range $[a, b]$. The Round(x) function maps each element of x to the nearest integer value, effectively quantizing continuous values into discrete levels. $\beta \in \mathbb{R}$ is defined as the average absolute value of the weight matrix, calculated by:

$$\beta = \max\left(\frac{1}{D_{in} D_{out}} \sum_{d_{in} d_{out}} |\mathbf{W}_{d_{in} d_{out}}|, \epsilon\right), \quad (2)$$

where D_{in}, D_{out} is the total number of elements in the weight matrix $\mathbf{W}$; d_{in} and d_{out} index these elements; and $|\mathbf{W}_{d_{in} d_{out}}|$ is the absolute value of each element. ϵ is a small constant added to prevent division by zero.

We define input matrix as $\mathbf{x} \in \mathbb{R}^{N \times D_{in}}$, where N is the number of tokens in a sample, and D_{in} represents the dimensionality of each token. To preserve the variance after quantization, previous research proposed introducing a LayerNorm (Ba et al., 2016) function before activation quantization.

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu(\mathbf{x})}{\sqrt{\text{Var}(\mathbf{x}) + \epsilon}} \quad (3)$$

Subsequently, the activation is quantized to 8-bit ($Q_b = 2^7$) using absmax quantization (Dettmers et al., 2022b). The input is scaled and quantized to form $\tilde{\mathbf{x}} \in \mathbb{R}^{N \times D_{in}}$, in which the ith row $\tilde{\mathbf{x}}_i$ is calculated as follows:

$$\tilde{\mathbf{x}}_i = \text{clip}\left(\text{Round}(\hat{\mathbf{x}}_i \times \frac{Q_b}{\gamma_i}), -Q_b, Q_b - 1\right), \quad (4)$$

$$\gamma_i = \max\left(\max_j(|\hat{\mathbf{x}}_{ij}|), \epsilon\right), \quad (5)$$

where $\gamma_i \in \mathbb{R}$ denotes scale factor for the ith token and $\hat{\mathbf{x}}_{ij}$ denotes the jth dimension of the ith token. We concatenate each scale factor $(\gamma_1, .., \gamma_i, .., \gamma_N)$ to form $\gamma \in \mathbb{R}^{N \times 1}$. We define $\mathbf{1}_{D_{out}} \in \mathbb{R}^{1 \times D_{out}}$ as a row vector of one. Finally, the output activation $\mathbf{y} \in \mathbb{R}^{N \times D_{out}}$ is rescaled for dequantization to the original precision.

$$\mathbf{y} = (\tilde{\mathbf{x}} \widetilde{\mathbf{W}} \odot \gamma \mathbf{1}_{D_{out}}) \times \frac{\beta}{Q_b} \quad (6)$$

3.3 BitEmbedding: Embedding layer Quantization Function

First, we describe a method for converting the embedding parameters into 1.58 bits for training. $\mathbf{W} \in \mathbb{R}^{V \times D}$ is an embedding matrix, with V being the vocabulary size and D the embedding dimension. $\mathbf{x} \in \mathbb{Z}^N$ represents the input sequence of token indices, where N is the sequence length. The embedding is subjected to ternary quantization.

$$\widetilde{\mathbf{W}} = \text{clip}\left(\text{Round}\left(\frac{\mathbf{W}}{\beta}\right), -1, 1\right), \quad (7)$$

where $\beta \in \mathbb{R}$ is a scaling factor that belongs to the category of token-wise normalization methods, ensuring that the embedding values are appropriately bounded. β is computed as follows:

$$\beta = \max\left(\frac{1}{VD} \sum_{vd} |\mathbf{W}_{vd}|, \epsilon\right). \quad (8)$$

In the equation, v indexes a specific word in the vocabulary, and d refers to the dimension within the embedding vector for that word. The term $\epsilon > 0$ is a small positive constant used to prevent numerical instability and ensure that β remains nonzero, even if all the elements of $\mathbf{W}_{vd}$ are small.

The final embedding representation $\mathbf{E}$ for the input sequence is formed by retrieving the vectors from the quantized matrix $\widetilde{\mathbf{W}}$ corresponding to the token indices in $\mathbf{x}$. An embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times D}$ is obtained using the following lookup operation:

$$\mathbf{E} = \widetilde{\mathbf{W}}[\mathbf{x}] \quad (9)$$

Subsequently, $\mathbf{E}$ undergoes layer normalization, which is formulated as:

$$\hat{\mathbf{E}} = \frac{\mathbf{E} - \mu(\mathbf{E})}{\sqrt{\mathrm{Var}(\mathbf{E}) + \epsilon}} \tag{10}$$

Thereafter, we describe a method for adjusting the embedded outputs into 8-bit. The scale factor for the ith token $\gamma_i \in \mathbb{R}$ is calculated as follows:

$$\gamma_i = \max\left(\max_j(|\hat{\mathbf{E}}_{ij}|), \epsilon\right) \tag{11}$$

Here, $\mathbf{E}_{ij}$ denotes the jth embedding dimension of the ith token. Subsequently, the embedding matrix is scaled and quantized. We quantize the embedding to an 8-bit ($Q_b = 2^7$) matrix $\widetilde{\mathbf{E}} \in \mathbb{R}^{N\times D}$, where the ith row $\widetilde{\mathbf{E}}_i$ is calculated as follows:

$$\widetilde{\mathbf{E}}_i = \mathrm{clip}\left(\mathrm{Round}(\hat{\mathbf{E}}_i \times \frac{Q_b}{\gamma_i}), -Q_b, Q_b - 1\right) \tag{12}$$

During forward propagation, the final output $\mathbf{E}_{out} \in \mathbb{R}^{N\times D}$ is computed as follows:

$$\mathbf{E}_{\mathrm{out}} = (\widetilde{\mathbf{E}} \odot \gamma \mathbf{1}_D) \times \left(\frac{\beta}{Q_b}\right), \tag{13}$$

where $\gamma \in \mathbb{R}^{N\times 1}$ is a concatenation of $(\gamma_1, .., \gamma_N)$ and $\mathbf{1}_D \in \mathbb{R}^{1\times D}$ is a row vector of ones. The above formula ensures that the quantization-aware training model has 1.58-bit weights for the embedding layer and 8-bit output. We employed a straight-through estimator (Bengio et al., 2013) to approximate the gradient during backpropagation.

4 Experiments

4.1 Setup

Models We pre-trained Mamba-2 models with sizes ranging from 80M to 170M parameters, employing the same pre-training methodology as a GPT-like autoregressive decoder (Brown et al., 2020). In particular, for the 170M model with a vocabulary size of 50288, the proportions of trainable parameters in various layers are listed in Table 1. The linear layers account for 53% of the parameters, whereas the language model head and embedding layers account for 47%. Meanwhile, the normalization and convolutional layers accounted for a relatively small proportion, and the SSM layers contained almost no trainable parameters, allowing them to be maintained at full precision. We explored three key model variants to focus on different quantization strategies. "Slender-Mamba-EQ" quantizes only the embedding layers. "Slender-Mamba-LQ" model applies quantization to all linear layers including a head layer, which is a slight different setting from the conventional BitNet. "Slender-Mamba" quantizes the linear layers as well as embedding and head layers to achieve a highly compact and efficient model.

Component	Parameters	Total (%)
conv1d layers	215,040	0.13
normalization	36,864	0.02
head layer	38,731,776	23.08
embedding	38,731,776	23.08
linear layers	90,095,616	53.68

Table 1: Trainable parameter distribution of each component for Mamba-2 (Model size 170M).

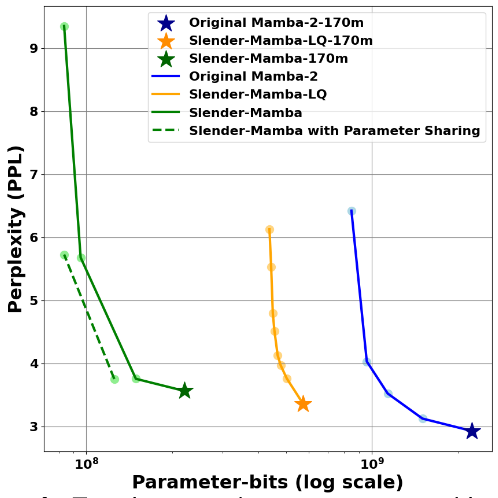

Figure 2: Frontier curve between parameter-bits and training loss. Each line is created by changing the number of layers (=model parameter size) for each model.

Settings Models on Figure 2 are pre-trained with 1B tokens using instruction data (Xu et al., 2024), employing a custom-trained tokenizer with a vocabulary size of 32000, which is comparatively small dataset owing to the computational limitation for experiments of several model sizes. Models on Table 2 are pre-trained with 150B tokens (4500 iterations) from FineWeb-Edu dataset (HuggingFaceFW, 2024). The models in Table 4 were pre-trained with 20B tokens (600 iterations) from the same dataset. These models used GPT-NEOX-20B tokenizer (Black et al., 2022) with a vocabulary of 50288. The context length was set to 4096 tokens. The experiments were conducted on 8 H100 GPUs, each with a batch size of 8. The gradient accumulation step was set at 128. The detailed training hyperparameters are described in Appendix B.

Evaluation For downstream tasks, we evaluate models by zero-shot performance, including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), ARC (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018).

Models	$P_b \times 10^8$	WG	PIQA	OBQA	HS	BoolQ	ARC-E	ARC-C	Avg.
Mamba-2	26.8	48.7	59.1	29.6	27.6	53.6	40.9	22.6	40.3
Slender-Mamba-EQ	21.2	51.3	**60.7**	28.2	**29.3**	61.5	**44.4**	24.2	**42.8**
Slender-Mamba-LQ	8.27	50.8	56.4	**30.8**	28.3	**61.7**	39.2	24.4	41.7
Slender-Mamba	2.69	**51.8**	55.6	27.0	27.7	58.5	38.6	**25.2**	40.6
Mamba-2-EQ	21.2	50.0	53.1	27.0	26.2	43.3	33.4	21.5	36.4
Mamba-2-LQ	8.27	49.1	50.1	24.6	26.2	40.7	27.9	24.2	34.7
Mamba-2-(E+L)Q	2.69	50.8	48.9	24.6	24.6	41.3	27.4	26.0	34.8

Table 2: Model performance on various benchmarks. HS, WG, and OBQA are abbreviations for HellaSwag, Winogrande, and OpenbookQA. Bold font indicates the best performance for the task in that column under pre-training, while an underline represents the best performance for the task in that column under direct quantization.

4.2 Results

Frontier Curve Between Parameter-Bits and Training Loss To quantitatively evaluate the trade-offs between model compression and pre-training performance as a result of BitNet quantization, we define "parameter-bits" (the total bits used by model parameters) as an evaluation metrics for compression. Specifically, the total number of parameter-bits P_b is calculated using the following equation: $P_b = \sum_{i=1}^{N_p} b_p^i$, where N_p is the total number of parameters and b_p^i is the number of bits per parameter for the i-th indexed parameter. This study quantizes the model parameters to 1.58-bit to ensure that $b_p = 1.58$ after BitNet quantization and $b_p = 16$ before quantization.

Figure 2 shows frontier curve between training loss (perplexity) and parameter-bits for each model. Each line is drawn by changing the total number of parameters (by changing the number of layers) for each model from 80M (number of layers:1) to 170M (number of layers:24). The result demonstrates that the frontier curve shifts to left side as the quantization ratio increases (Original Mamba-2 -> Slender-Mamba-LQ -> Slender-Mamba), indicating that we could reach a new frontier curve by our quantization method, which cannot be achieved by normal hyper-parameter tuning. Additionally, we conducted experiments by applying the parameter-sharing method to Slender-Mamba, achieving the most left-sided frontier curve. For reference, we conducted additional experiments by applying the parameter-sharing method to Slender-Mamba to further reduce the model size. This result yields the most left-sided frontier curve. Although the frontier curve improved, there was a noticeable degradation in the performance of downstream tasks. The detailed performance of the Parameter Sharing method is described in Appendix C.

Performance of Downstream Tasks We tested the performance of four architectures: Slender-Mamba-EQ, Slender-Mamba-LQ, Slender-Mamba, and the original Mamba-2 on downstream tasks. We pre-trained the four models (170M) using the same data and parameters, with the detailed training parameters provided in Appendix B. We followed the pipeline from lm-evaluation-harness (Gao et al., 2024) to perform the evaluation.

As shown in Table 2, pre-trained models using our method (Slender-Mamba series) showed no significant performance degradation on downstream tasks while achieving a reduction of parameter-bits. Especially, Slender-Mamba achieves 90.0% reduction in parameter-bits ($2.68 \cdot 10^9 \rightarrow 2.69 \cdot 10^8$) while maintaining the competitive performance as original Mamba-2. Similar to the results of BitNet1.58 (Ma et al., 2024b), our approach, which pre-trains models using Quantization-Aware Training (QAT), can maintain a performance on par with full-precision models and occasionally surpasses them owing to performance fluctuations. In contrast, Mamba-2-EQ, Mamba-2-LQ, and Mamba-2-(E+L)Q directly apply 1.58-bit quantization to the original pre-trained Mamba-2 model for the embedding layer, linear layer, and both embedding and linear layers, respectively. Under the same parameter-bits, their scores were lower than those of the corresponding Slender–Mamba models, with the average score decreasing linearly as the parameter-bits were reduced. Appendix D describes the performance of the other baselines. Appendix E describes embedding layer analysis.

5 Conclusion

This study introduced Slender-Mamba, a highly parameter-efficient quantized language model that further advances the lightweight design of Mamba. We extended BitNet quantization method, which generally focuses on linear layers, to include embedding and projection layers. The proposed method achieved a significant reduction of bits used by the model parameters with minimal performance degradation. We hope that these findings will provide the potential to incorporate more lightweight language models into edge devices, which will become more demanding in the future.

6 Limitations

This study has several limitations. First, owing to time constraints, we could not further verify the performance of our model when the embeddings were quantized to 1-bit. Although the preliminary results are promising, additional experiments would be beneficial to fully ascertain the impact of such extreme quantization on the model accuracy and stability. Second, our computational resources were limited, which restricted our ability to scale the model and expand the training dataset. Consequently, the findings may not fully represent the potential of the model when trained under more resource-abundant conditions. Future research with enhanced computational capabilities can thoroughly explore these aspects, potentially leading to more robust conclusions.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Anonymous. 2024. Mambaquant: Quantizing the mamba family with variance aligned rotation methods. In Submitted to The Thirteenth International Conference on Learning Representations. Under review.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. Preprint, arXiv:1607.06450.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 7432–7439.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. arXiv preprint arXiv:2204.06745.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.

Xiangxiang Chu, Jianlin Su, Bo Zhang, and Chunhua Shen. 2024. Visionllama: A unified llama interface for vision tasks. arXiv preprint arXiv:2403.00522.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. arXiv preprint arXiv:1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. arXiv preprint arXiv:2405.21060.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. arXiv preprint arXiv:1807.03819.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022a. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. Advances in Neural Information Processing Systems, 35:30318–30332.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022b. Llm.int8(): 8-bit matrix multiplication for transformers at scale. Preprint, arXiv:2208.07339.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. 2022. Hungry hungry hippos: Towards language modeling with state space models. arXiv preprint arXiv:2212.14052.

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. 2024. A framework for few-shot language model evaluation.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. Advances in neural information processing systems, 33:1474–1487.

HuggingFaceFW. 2024. fineweb-edu (revision 22b0aca).

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. arXiv preprint arXiv:2402.14905.

Liqun Ma, Mingjie Sun, and Zhiqiang Shen. 2024a. Fbi-llm: Scaling up fully binarized llms from scratch via autoregressive distillation. arXiv preprint arXiv:2407.07093.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024b. The era of 1-bit llms: All large language models are in 1.58 bits. arXiv preprint arXiv:2402.17764.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In EMNLP.

Libo Qin, Qiguang Chen, Yuhang Zhou, Zhi Chen, Yinghui Li, Lizi Liao, Min Li, Wanxiang Che, and Philip S Yu. 2024. Multilingual large language model: A survey of resources, taxonomy and frontiers. arXiv preprint arXiv:2404.04925.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. Communications of the ACM, 64(9):99–106.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 13693–13696.

Sho Takase and Shun Kiyono. 2021. Lessons on parameter sharing across layers in transformers. arXiv preprint arXiv:2104.06022.

A Vaswani. 2017. Attention is all you need. Advances in Neural Information Processing Systems.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. arXiv preprint arXiv:2310.11453.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning, pages 38087–38099. PMLR.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In The Twelfth International Conference on Learning Representations.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? arXiv preprint arXiv:1905.07830.

Rui-Jie Zhu, Yu Zhang, Ethan Sifferman, Tyler Sheaves, Yiqiao Wang, Dustin Richmond, Peng Zhou, and Jason K Eshraghian. 2024. Scalable matmul-free language modeling. arXiv preprint arXiv:2406.02528.

A Related Work Continued

The introduction of a baseline model called MobileLLM (Liu et al., 2024) demonstrated superior performance over existing sub-billion models through architectural innovations such as deep and thin layers, embedding sharing, and grouped-query attention mechanisms. Previous studies (Takase and Kiyono, 2021) have shown that parameter-sharing strategies can significantly reduce the number of parameters required by a model while maintaining or even enhancing its performance. This is particularly critical for large-scale language models, as it helps mitigate computational and storage challenges.

B Training parameters

Table 3 lists the training hyperparameters of Slender Mamba for the experiments.

Parameter	**Value**
model dimension	768
number of layers	24
Optimizer	AdamW
learning rate schedule	cosine
Learning Rate	1.6×10^{-3}
AdamW Betas	(0.9, 0.95)
Weight Decay	0.1
Warmup Ratio	0.1
Batch Size	8192

Table 3: Optimizer and Training Parameters

C Parameter Sharing

In our study, we followed the methodology of previous research by applying various parameter-sharing strategies to optimize the performance of Mamba models. These strategies, including Sequential Sharing (SEQUENCE), Cyclical Sharing (CYCLE), Reverse Cyclical Sharing (CYCLE (REV)), and Universal Sharing (UNIVERSAL) (Dehghani et al., 2018), are described in Algorithm 1. In our experiments, we focused on the performance of these parameter-sharing strategies across different sizes and configurations of Mamba models. Notably, Sequential Sharing (SEQUENCE), Cyclical Sharing (CYCLE), and Reverse Cyclical Sharing (CYCLE (REV)) strategies effectively reduced the total number of parameters by approximately 50% while maintaining the total number of layers. This significant parameter reduction decreases the computational complexity as well as preserves the expressiveness of the model. In particular, the Universal Sharing (UNIVERSAL) strategy represents an extreme case of parameter compression that condenses the parameters of all the layers into a single layer. While this approach maximizes parameter efficiency, it may present challenges, such as potential loss in model expressiveness. We conducted pre-training tests using the Parameter Sharing method in the Slender-Mamba architecture to further reduce total parameter-bits. The results of the downstream task tests are presented in Table 4.

Algorithm 1 Parameter Sharing Strategy

Input:
Total number of layers, (N) Number of independent layers (M) **Output**: $\text{enc}_1, \ldots, \text{enc}_N$

```
 1: for $i = 1$ to $N$ do
 2:     if $i == 1$ then
 3:         $\text{enc}_i \leftarrow \text{CreateNewLayer}()$
 4:     else if TYPE == SEQUENCE then
 5:         if $(i-1) \mod \lfloor N/M \rfloor == 0$ then
 6:             $\text{enc}_i \leftarrow \text{CreateNewLayer}()$
 7:         else
 8:             $\text{enc}_i \leftarrow \text{enc}_{i-1}$
 9:         end if
10:     else if TYPE == CYCLE then
11:         if $i \leq M$ then
12:             $\text{enc}_i \leftarrow \text{CreateNewLayer}()$
13:         else
14:             $\text{enc}_i \leftarrow \text{enc}_{((i-1) \mod M)+1}$
15:         end if
16:     else if TYPE == CYCLE(REV) then
17:         if $i \leq M$ then
18:             $\text{enc}_i \leftarrow \text{CreateNewLayer}()$
19:         else if $i \leq (M \times (\lceil N/M \rceil - 1))$ then
20:             $\text{enc}_i \leftarrow \text{enc}_{((i-1) \mod M)+1}$
21:         else
22:             $\text{enc}_i \leftarrow \text{enc}_{M-((i-1) \mod M)}$
23:         end if
24:     else if TYPE == UNIVERSAL then
25:         $\text{enc}_i \leftarrow$ shared_block
26:     end if
27: end for
```

D Performance on downstream tasks compared to other baselines

We compared our work with ternary quantization LLM of similar size, BitNet b1.58 (Ma et al., 2024b) and binarized LLM FBI-LLM (Ma et al., 2024a), which use knowledge distillation. We further included results from open-source full-precision models of similar sizes, such as MobileLLM (Liu et al., 2024), Mamba (Gu and Dao, 2023), and Mamba-2 (Dao and Gu, 2024). We compared our work with recent studies that apply post-training quantization (PTQ) at 8-bit and 4-bit to the Mamba model, as detailed in MambaQuant (Anonymous, 2024). BW means bit-width occupied by model parameters, excluding the embedding layer and the head. The results, as shown in Table 5, indicate that although our model has a bit-width (BW) of 1.59, the overall parameter-bits are significantly reduced owing to the quantization of both the linear and embedding layers. Moreover, our model nearly matches the performance level of the baseline model with only pre-training on the 150B dataset, which sufficiently demonstrates the effectiveness of our proposed method. Compared to the Transformer architecture, the application of BitNet to Mamba shows only a small gap and even surpasses the baseline, demonstrating the effectiveness of BitNet under the SSMs framework.

E Embedding Matrices Analysis

The objective of this study is to ascertain whether word embeddings experience minimal or no degradation after conversion to BitNet format. The experiment aimed to confirm that the distances between words within word embeddings did not undergo significant changes before and after conversion to BitNet. Word-embedding matrices were prepared from the four models described in the ex-

Models	$P_b \times 10^8$	WG	PIQA	OBQA	HS	BoolQ	ARC-E	ARC-C	Avg.
Slender-Mamba-SEQUENCE	1.97	48.6	51.3	**26.6**	**26.6**	37.8	26.1	**26.5**	34.8
Slender-Mamba-CYCLE	1.97	48.9	**53.4**	24.8	24.9	37.8	28.5	25.3	34.8
Slender-Mamba-CYCLE(REV)	1.97	49.4	53.3	26.4	24.4	38.1	**32.3**	22.4	35.2
Slender-Mamba-Universal	1.32	**50.3**	52.0	24.6	25.0	**56.3**	27.6	21.9	**36.8**

Table 4: Model applying parameter-sharing technique performance on various benchmarks. HS, WG, and OBQA are abbreviations for HellaSwag, Winogrande, and OpenbookQA, respectively. Bold font indicates the best performance for the task in that column under pre-training

Model	Architecture	BW	data size	WG	PIQA	OBQA	HS	BoolQ	ARC-E	ARC-C	Avg.
MobileLLM-125M	Transformer	16	1T	53.1	65.3	**39.5**	**38.9**	60.2	43.9	**27.1**	**46.9**
Mamba-130M	Mamba	16	300B	52.6	63.1	28.6	35.2	55.0	41.9	24.2	43.0
Mamba-2-130M	Mamba-2	16	300B	52.1	64.1	30.6	35.3	55.2	42.0	24.2	43.3
MambaQuant-130M-W8A8	Mamba	8	300B	52.3	62.5	-	34.7	-	45.9	24.2	43.9
MambaQuant-130M-W4A8	Mamba	4	300B	50.9	56.9	-	30.5	-	36.2	24.6	39.8
BitNet b1.58-700M	Transformer	1.59	100B	**55.2**	**68.0**	20.0	35.1	58.2	**51.8**	21.4	44.2
FBI-LLM-130M	Transformer	1	1.2T	51.0	59.3	26.4	28.7	**62.1**	34.9	20.5	40.4
Slender-Mamba-170M(Ours)	Mamba-2	1.59	150B	51.8	55.6	27.0	27.7	58.5	38.6	25.2	40.6

Table 5: Performance on downstream tasks compared to other baselines. BW means bit-width occupied by model parameters, excluding the embedding layer and the head. HS, WG, and OBQA are abbreviations for HellaSwag, Winogrande, and OpenbookQA, respectively. Bold font indicates the best performance for the task in that column under pre-training

perimental results. Table 2. The word-embedding matrices $\mathbf{W}_1$ for the original Mamba-2, $\mathbf{W}_2$ for Slender-Mamba-LQ, $\mathbf{W}_3$ for Slender-Mamba, $\mathbf{W}_4$ for Mamba-2-EQ and $\mathbf{W}_5$ for Slender-Mamba-EQ. Subsequently, the cosine distances among all the words in the embeddings were computed.

$$\text{cosine\_similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \tag{14}$$

Here, $\mathbf{X}_n$ represents the cosine similarity matrix for the embedding matrix $\mathbf{W}_n$, where each element $\mathbf{X}_n[i, j]$ corresponds to the cosine similarity between the ith and jth column vectors of $\mathbf{W}_n$. This metric quantifies semantic relationships within the embedding space of the specified model.

$$\mathbf{X}_n[i, j] = \text{cosine\_similarity}(\mathbf{w}_n^i, \mathbf{w}_n^j), \tag{15}$$

where $\mathbf{w}_n^i$ and $\mathbf{w}_n^j$ are the ith and jth column vectors of $\mathbf{W}_n$ and $n = 1, 2, 3, 4, 5$.

Subsequently, the degree of degradation was measured using the $L2$ distance, which calculates the Euclidean distance between the two matrices $\mathbf{X}_1$ and $\mathbf{X}_n$. This distance quantifies the dissimilarity introduced during quantization or other modifications, thereby reflecting the extent of deviation from the original embedding matrix.

$$L2\_distance(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{16}$$

$$\mathbf{Y}_{(1,n)} = L2\_distance(\mathbf{x}_1, \mathbf{x}_n) \tag{17}$$

where $n = 2, 3, 4, 5$, $\mathbf{x}_1 = \mathbf{X}[1, :]$ represents the first row vector of $\mathbf{X}$ and $\mathbf{x}_n = \mathbf{X}[n, :]$ denotes the n-th row vector. $\mathbf{Y}_{(1,n)}$ represents the computed degradation measure for each pair of the original embedding matrix $\mathbf{X}_1$ and its modified counterparts $\mathbf{X}_n$. The results are listed in Table 6. Among these, $\mathbf{X}_2$, $\mathbf{X}_3$, and $\mathbf{X}_5$ represent the proposed methods, whereas $\mathbf{X}_4$ involves the direct quantization of the embedding layer in Mamba-2, which does not align with the results in Table 2. The impact of varying degrees of BitNet application on the word embeddings of the Mamba model was examined by calculating and comparing the cosine similarity and Euclidean distance between these embeddings. The performance degradation owing to model changes was assessed. Furthermore, the correlation coefficients for the embedding layers of Slender-Mamba-EQ and Slender-Mamba compared with the original Mamba-2 are relatively small. Figure 3 shows a scatter plot of the embedding comparisons across various configurations: $\mathbf{X}_1$ versus $\mathbf{X}_2$ (top left), $\mathbf{X}_1$ versus $\mathbf{X}_3$ (top right), $\mathbf{X}_1$ versus $\mathbf{X}_4$ (bottom left), and $\mathbf{X}_1$ versus $\mathbf{X}_5$ (bottom right). The comparisons revealed distinct patterns in the embeddings depending on the quantization and replacement strategies employed. Considering these observations, it is hypothesized that the gradient propagation method inherent in BitEmbedding differs fundamentally from unquantized approaches. However, according to the results in Table 2 the implementation of BitEmbedding does not result in degradation.

Comparison	L2 Distance	Correlation
$\mathbf{Y}_{1,2}$ ($\mathbf{X}_1$ vs. $\mathbf{X}_2$)	1694.1	0.006
$\mathbf{Y}_{1,3}$ ($\mathbf{X}_1$ vs. $\mathbf{X}_3$)	2071.3	0.498
$\mathbf{Y}_{1,4}$ ($\mathbf{X}_1$ vs. $\mathbf{X}_4$)	791.7	0.894
$\mathbf{Y}_{1,5}$ ($\mathbf{X}_1$ vs. $\mathbf{X}_5$)	2083.2	0.0387

Table 6: L2 Distances and Correlation Coefficients Between Word Embedding Matrices Before and After BitNet Conversion

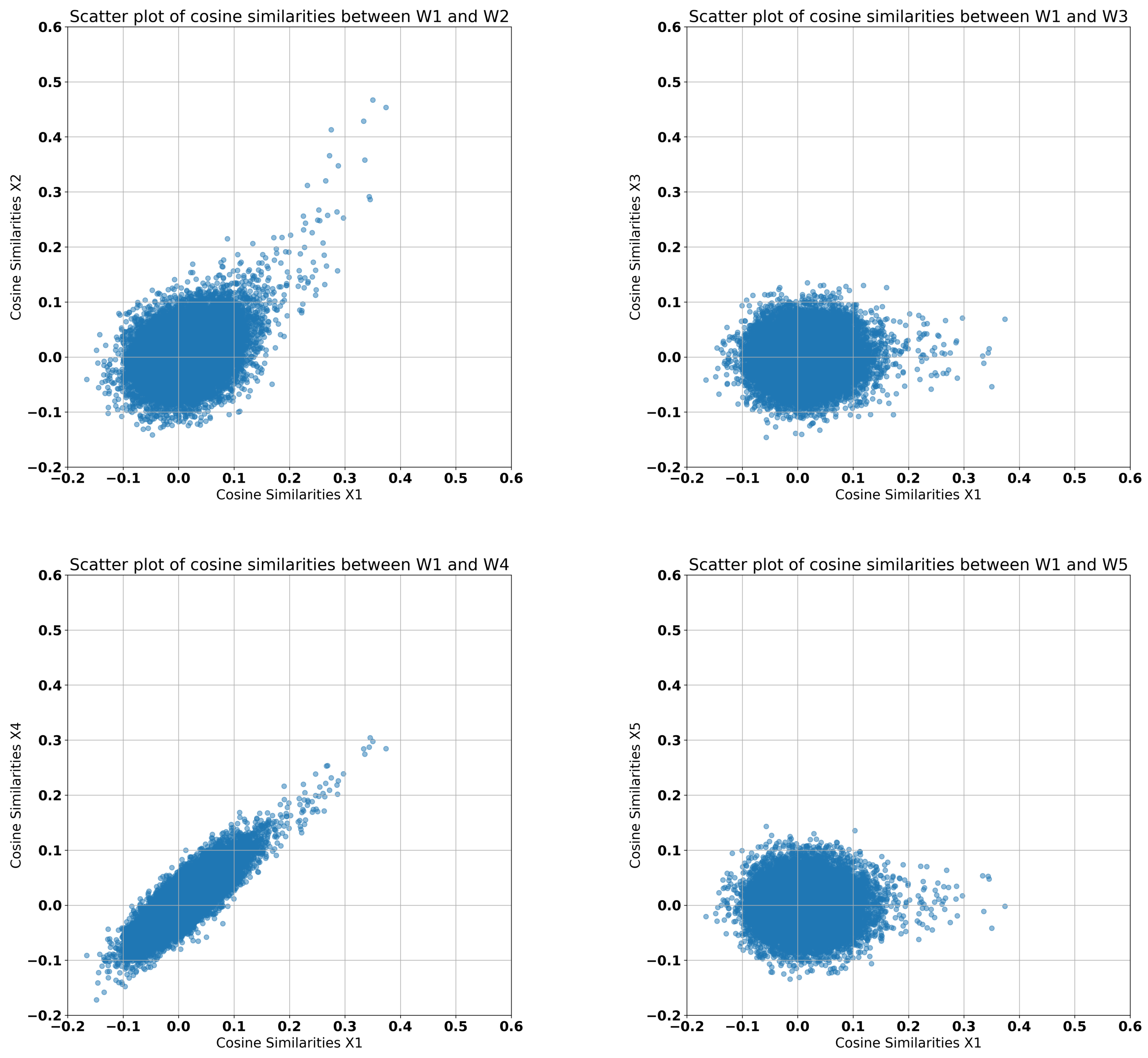

Figure 3: Scatter plot distributions of embedding comparisons: $\mathbf{X}_1$ versus $\mathbf{X}_2$ (top left), $\mathbf{X}_1$ versus $\mathbf{X}_3$ (top right), $\mathbf{X}_1$ versus $\mathbf{X}_4$ (bottom left), and $\mathbf{X}_1$ versus $\mathbf{X}_5$ (bottom right). $\mathbf{X}_1$ does not use BitEmbedding as a replacement for the embedding layer, whereas $\mathbf{X}_4$ directly quantizes the embedding layer, showing some correlation. In contrast, $\mathbf{X}_3$ and $\mathbf{X}_5$, which replace the embedding layer with BitEmbedding, do not exhibit such correlation.