QUAMBA: A POST-TRAINING QUANTIZATION RECIPE FOR SELECTIVE STATE SPACE MODELS

Anonymous authors
Paper under double-blind review

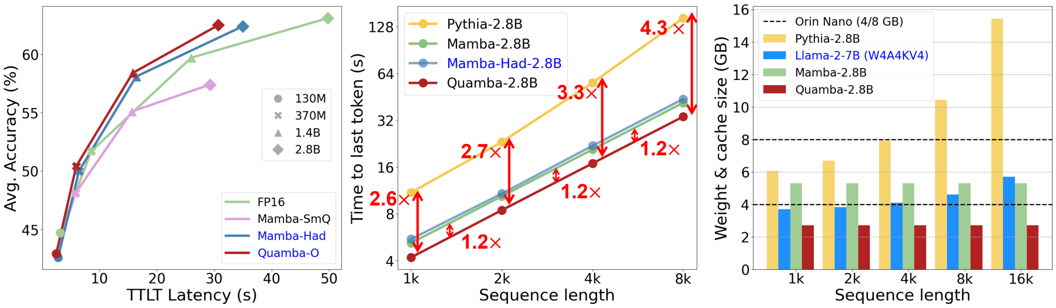
ABSTRACT

State Space Models (SSMs) have emerged as an appealing alternative to Transformers for large language models, achieving state-of-the-art accuracy with constant memory complexity which allows for holding longer context lengths than attention-based networks. The superior computational efficiency of SSMs in long sequence modeling positions them favorably over Transformers in many scenarios. However, improving the efficiency of SSMs on request-intensive cloud-serving and resource-limited edge applications is still a formidable task. SSM quantization is a possible solution to this problem, making SSMs more suitable for wide deployment, while still maintaining their accuracy. Quantization is a common technique to reduce the model size and to utilize the low bit-width acceleration features on modern computing units, yet existing quantization techniques are poorly suited for SSMs. Most notably, SSMs have highly sensitive feature maps within the selective scan mechanism (*i.e.,* linear recurrence) and *massive outliers* in the output activations which are not present in the output of token-mixing in the self-attention modules. To address this issue, we propose a *static* 8-bit per-tensor SSM quantization method which suppresses the maximum values of the input activations to the selective SSM for finer quantization precision and quantizes the output activations in an outlier-free space with Hadamard transform. Our 8-bit weight-activation quantized Mamba 2.8B SSM benefits from hardware acceleration and achieves a 1.72 × lower generation latency on an Nvidia Orin Nano 8G, with only a 0.9% drop in average accuracy on zero-shot tasks. When quantizing Jamba, a 52B parameter SSM-style language model, we observe only a 1% drop in accuracy, demonstrating that our SSM quantization method is both effective and scalable for large language models, which require appropriate compression techniques for deployment. The experiments demonstrate the effectiveness and practical applicability of our approach for deploying SSM-based models of all sizes on both cloud and edge platforms.

1 INTRODUCTION

State Space Models (SSMs) (Gu & Dao, 2023; Lieber et al., 2024b) have attracted notable attention due to their efficiency in long sequence modeling and comparable performance to Transformers (Zhang et al., 2022; Brown et al., 2020). Although Transformers have shown a strong ability to capture the causal relationships in long sequences, the self-attention module within Transformers incurs a quadratic computation complexity with respect to the context length in the prefilling stage as well as a linear memory complexity (*e.g.,* the K-V cache) in the generation stage. In contrast, SSMs, an attractive substitute to Transformers, perform sequence modeling with a recurrent neural network-like (RNN-like) linear recurrence module (*i.e.,* selective scan) which has *linear* computation complexity in the prefilling stage and *constant* memory complexity in the generation stage.

Despite their computational advantages, deploying SSMs on diverse hardware is challenging due to memory and latency constraints. Quantization, such as 8-bit integer (INT8), offers a solution by reducing model size and latency while preserving accuracy and enabling hardware acceleration. Quantizing SSMs is non-trivial as current post-training quantization (PTQ) techniques for Transformers (Xiao et al., 2023; Dettmers et al.) fail to handle the sensitive activations of the selective scan (*i.e.*, linear recurrence) resulting in poor performance (Lieber et al., 2024b; Zhao et al., 2024). Our study shows SSMs exhibit *distinct* outlier patterns compared to Transformers (Ashkboos et al., 2024b; Dettmers et al.; Xiao et al., 2023; Zhao et al., 2023). We show that outliers appear in the SSMs output

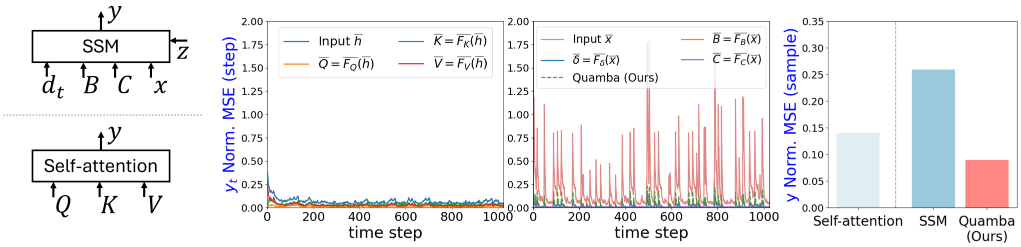


(a) Pareto-optimality on Nano 8G (b) Long input sequences on A5000 (c) Memory usage

Figure 1: We demonstrate that (a) our method achieves Pareto-optimality on the Nano 8G with 1K input tokens. Figure (b) shows latency speedups for long input sequences on the A5000, and Figure (c) shows the memory usage across devices comparing to Pythia 2.8B (Biderman et al., 2023) and 4-bit Llama-2-7B (Touvron et al., 2023).

(*i.e.* the y tensor). In contrast, the input and output of self-attention layers are relatively *smooth* and *do not* exhibit outlier issues. More comparisons can be found in Section K. This underscores the need for specialized quantization methods for SSM-based models, as their lack leads to suboptimal memory usage and latency on deployed hardware.

We first analyze the input and output activations of the selective scan module to reveal the quantization sensitivity and outliers in SSM activations (*ref.* Figure 2 and Figure 3). Specifically, we show that quantizing SSMs is particularly challenging since SSM input and output activations present a causal relationship, making the input tensor (*i.e.,* x in Eq. 1) sensitive to quantization errors. Additionally, representing the large outliers using 8-bit precision in the output activations (*i.e.,* y in Eq. 1), which are *not* present in the token-mixing output of self-attention modules, is difficult. To address this, we propose Quamba, an 8-bit *static per-tensor* quantization method for selective SSMs that can leverage the low bit-width acceleration features on modern computing units with *minimal overhead*. We suppress maximum values in input activations to SSMs, which is the most sensitive to the quantization error, for finer quantization precision. For the extreme outliers in output activations from SSMs, we use the Hadamard transform to smooth out the activations. Our quantized 8-bit 2.8B Mamba SSM achieves 1.72× speedup in generation latency (*i.e.,* time-per-output-token, TPOT) on Nvidia Orin Nano 8G while only incurring a 0.9% accuracy drop in zero-shot tasks. We demonstrate our method achieves Pareto-optimality on the Nano 8G and lower the latency by 1.2 × on A5000, as shown in Figure 1. While quantizing Jamba, a 52B parameter SSM-style language model, we observe just a $\sim 1\%$ reduction in accuracy. The effectiveness and scalability of our quantization technique for



(a) SSM *vs.* Self-attention (b) Quantizing self-attention (c) Quantizing SSM (d) Quantization errors

Figure 2: We analyze the sensitivity of quantization errors for (b) self-attention layers and (c) SSMs input activations. Our study shows that the x tensor causes huge errors at the output y due to the causal relationship of the linear recurrence, which is unique to SSMs. Self-attention layers are more robust to quantization errors. Our method (d) reduces the quantization error for the input sample. In Figure 13, we highlight the smooth, outlier, and sensitive paths in SSMs and self-attention layers.

SSM-based models, as well as the practicality of our approach for deploying SSM-based models of various sizes on cloud and edge platforms.

2 RELATED WORK

Model quantization. Deploying large, over-parameterized neural networks on resource-constrained devices is challenging, and researchers have developed model quantization (Han et al., 2015; Jacob et al., 2018; Wang et al., 2019) as a solution to this problem. Quantization techniques reduce the data type precision (*e.g.* FP32 to INT4) to compress the model size and accelerate inference. The quantization techniques are generally divided into two categories: Quantization-aware training (QAT) and post-training quantization (PTQ) (Zhu et al., 2023; Gholami et al., 2022; Zhou et al., 2024). QAT (Liu et al., 2023; Dettmers et al., 2024) requires additional training efforts to adapt models to low bit-width, in exchange for better model performance. Our work falls under PTQ, which does not require training and can therefore be plug-and-play.

LLM post-training quantization. Post-training quantization (PTQ) techniques are generally broken down into two categories: weight-only quantization (*e.g.,* W4A16 and W2A16) and weight-activation quantization (*e.g.,* W8A8 and W4A4) (Zhu et al., 2023). Weight-only quantization (Frantar et al., 2022; Lin et al., 2023) focuses on quantizing weight matrices (*e.g.,* 4-bit or 2-bit) while keeping the activations in half-precision floating point. However, although weight-only quantization reduces memory usage, it still requires costly floating-point arithmetic for compute-intensive operations (*e.g.,* linear layers). To utilize low bit-width operations, Xiao et al. (2023); Zhao et al. (2023); Ashkboos et al. (2024b); Dettmers et al. study quantization for both weights and activations in Transformers. They address outliers in activations by using mixed-precision (Dettmers et al.), rescaling quantization factors (Xiao et al., 2023), group quantization (Zhao et al., 2023), and quantizing activations in an outlier-free space (Ashkboos et al., 2024b). Unfortunately, these techniques target Transformers, do not generalize well to SSMs, and either fail to handle the sensitive tensors in SSMs resulting in poor performance (Xiao et al., 2023; Dettmers et al.) or introduce additional computational overhead to the input of the selective scan (Zhao et al., 2023; Ashkboos et al., 2024b). Our research addresses this gap by examining SSM weight-activation quantization, aiming to concurrently reduce memory and compute costs by harnessing hardware acceleration for integer operations.

State Space Models. In recent times, a new wave of RNN-like models (Gu et al., 2021; Smith et al., 2022; Peng et al., 2023; Gu & Dao, 2023; Beck et al., 2024) has emerged, noted for their efficacy in modeling long-range dependencies and achieving performance comparable to Transformers. State Space Models (SSMs) (Gu et al., 2021; Smith et al., 2022; Gu & Dao, 2023) are a promising class of architectures that have been successfully applied to various applications, such as text (Gu & Dao, 2023; Wang et al., 2024), image (Zhu et al., 2024; Nguyen et al., 2022), video (Li et al., 2024; Nguyen et al., 2022), and audio (Goel et al., 2022; Saon et al., 2023). Despite their successes, the challenge of deploying SSMs across resource-limited hardware platforms remains largely unresolved. Our work addresses this challenge by proposing a quantization method specifically tailored for SSMs.

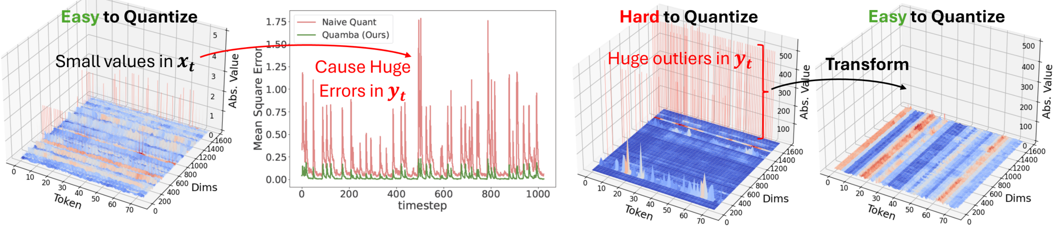
3 BACKGROUND

3.1 SELECTIVE STATE SPACE MODELS

State Space Models. Inspired by continuous systems, discrete linear time-invariant (LTI) SSMs (Gu et al., 2020; 2021; 2022) map input sequences $\mathbf{x}$ to output sequences $\mathbf{y}$. Given a discrete input signal x_t at time step t, the transformation $x_t \mapsto y_t$ through a hidden state h is defined as

$$h_t = \dot{A}h_{t-1} + \dot{B}x_t, \quad y_t = Ch_t + Dx_t \tag{1}$$

where $\dot{A}$ and $\dot{B}$ are discrete parameters. The discretization function for $\dot{A}$ and $\dot{B}$ with a given Δ is defined as $\dot{A} = \exp(\Delta A)$, $\dot{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B \approx \Delta B$. This system uses A as a state transition parameter and B and C as projection parameters. Δ is the time-scale parameter that is used to discretize the continuous parameters A and B. D is an optional residual parameter.



(a) SSM Input x_t (b) SSM Output y_t

Figure 3: The primary difficulties in quantizing Mamba blocks lie in the precision of the activations input into and output from the selective SSM. Although inputs are numerically small, the quantization step is skewed by the maximum value, causing significant errors in the output SSMs after the linear recurrent system. In contrast, large outliers are observed in the outputs. We use Hadamard matrices to transform the outputs to an outlier-free space.

(A, B, C, D, Δ) are trainable parameters. A residual branch z_t is applied to the SSM output such that $y_t \cdot \text{SiLU}(z_t)$ before the output projection.

SSMs with selection. Gu & Dao (2023) improve SSMs with selection by letting their parameters B, C, and Δ be input-dependent, allowing the model to selectively remember or ignore inputs based on their content. Specifically, the interaction with input x_t is defined as $B_t = \text{F}_B(x_t), \quad C_t = \text{F}_C(x_t), \quad \Delta_t = \text{softplus}(\text{F}_\Delta(x_t))$ where F_B and F_C are linear layers that map $x_t \mapsto B_t, C_t$. F_Δ use two consecutive projection layers, such that $\text{F}_\Delta = \text{Proj}(\text{Proj}(x)) + \text{bias}$. With the selection mechanism, the model has changed from time-invariant to time-varying.

3.2 QUANTIZATION

We focus on symmetric uniform quantization to approximate floating-point weights and activations with discrete 8-bit signed integers (*i.e.,* INT8) due to its hardware compatibility. The general symmetric uniform quantization function is defined as

$$\overline{X} = \text{clamp}\Big(\Big\lfloor \frac{X}{s} \Big\rceil, -2^{N-1}, 2^{N-1} - 1\Big), \quad s = \frac{\max\big(|X|\big)}{2^{N-1} - 1}, \tag{2}$$

where $\overline{X}$ represents the quantized weights or activations in INT8, X is the input matrix in floating point, and s is the scaling factor (*i.e.,* quantization step) that is determined by the target bit-width N ($N = 8$ in our setting). The *static* scaling factor s is pre-calibrated on a subset of data and is *fixed* during inference. We use the notation X to represent the floating-point matrices, and $\overline{X}$ to represent their quantized matrices with their floating-point scaling factors s_x. For operators, we use $\overline{f}(\cdot)$ to represent the quantized version of the function $f(\cdot)$ (*i.e.,* the weights are quantized in the function $\overline{f}$).

3.3 WALSH–HADAMARD TRANSFORM

Hadamard matrices. A Hadamard matrix is an n-dimensional square matrix whose entries are either $+1$ or -1, and the rows and columns are mutually orthogonal with the computational property $\mathbf{H}_n\mathbf{H}_n^\top = n\mathbf{I}_n$. Walsh-Hadamard matrix is a special category of Hadamard matrix, consisting of square matrices of size 2^k and can be constructed as follows:

$$\mathbf{H}_{2^k} = \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & -\mathbf{H}_{2^{k-1}} \end{bmatrix} = \mathbf{H}_2 \otimes \mathbf{H}_{2^{k-1}} \quad \text{, where} \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Walsh–Hadamard transform. The Walsh–Hadamard transform (WHT), a generalized class of Fourier transforms, has been applied to many related areas, such as LLM quantization (Ashkboos et al., 2024b) and efficient transfer learning (Yang et al., 2024), due to its favorable computational properties. We perform WHT to remove outliers from the output of the selective SSM. WHT projects a discrete input sequence signal onto a set of square waves (*i.e.,* Walsh functions). Its forward and inverse transform can be expressed in matrix form as $\tilde{x} = \mathbf{H}_n x$, and $x = \mathbf{H}_n^\top \tilde{x}$. x is the input discrete sequence signal, and $\tilde{x}$ denotes the WHT coefficients (*i.e.,* sequence components) that describe the magnitude of the square waves. WHT is efficient since the transform matrices consist only of real numbers, +1 or -1, so no multiplications are needed. The fast Walsh–Hadamard transform (FWHT) can compute the transformation with a complexity of $n\log n$ in a GPU-friendly (*i.e.,* parallelizable) fashion (Dao, 2024b). For input dimension $n \neq 2^k$, we factorize $n = 2^p m$, where m is the size of a known Hadamard matrix (Sloane, 1999).

4 QUAMBA: QUANTIZING MAMBA BLOCKS

4.1 PRELIMINARY STUDY

Theoretical error bound for SSM quantization. We consider a discrete linear time-invariant (LTI) state space model at time step t defined as $h(t) = A(t)h(t-1) + Bx(t)$, where the state matrix $A(t) \in \mathbf{R}^{N\times N}$, input matrix $B \in \mathbf{R}^{N\times P}$, implicit latent state $h(t) \in \mathbf{R}^{N\times 1}$, and input $x(t) \in \mathbf{R}^{P\times 1}$. We assume the norm of the state matrix A can be bounded by an exponential function such that $||A(t)||_2 \leq a \cdot e^{t-T}$, and the input matrix B is bounded by $||B|| \leq b$, where $0 < a < 1$, $b > 0$, and $1 \leq t \leq T$. In our proof, we utilize the spectral norm ($|| \cdot || = || \cdot ||_2$). Furthermore, we assume the system input contains a quantization error $\delta_x(t) \in \mathbf{R}^{P\times 1}$, such that $\overline{x}(t) = x(t) + \delta_x(t)$, where $||\delta_x(t)||_2 \leq \epsilon$ and $\epsilon > 0$. The system is initialized as $h(0) = [0, 0, \ldots 0]^T$. For details of the proof, please check Section H.2 in Appendix H.

Theorem 4.1. *The quantization error of* $\Delta(t) = \overline{h}(t) - h(t)$ *at time step t for the given discrete linear time-invariant model is bounded such that:* $||\Delta(t)||_2 \leq \epsilon b\left(\frac{1}{1-ae^{t-T}}\right)$. *Consequently, the* ***global*** *quantization error* (i.e., $t = T$) *is bounded by :* $||\Delta(T)||_2 \leq \frac{\epsilon b}{1-a}$.

Empirical analysis of SSM quantization. As shown in Figure 2 and Figure 3, the main challenge in quantizing Mamba blocks is the precision of the SSM input activation x, which is sensitive to quantization errors, and the output activation y. In Figure 1 (a), the naive 8-bit quantization introduces large quantization errors, resulting in model collapse for all model sizes. We delve into the causal relationship between (A, B, C, Δ, x) and y as modeled by the linear recurrent system in Equation 1. As shown in Figure 2 (b) and Figure 3 (a), we find that x is sensitive to quantization errors and it leads to the largest errors in the SSM output y, although the values in the x tensor are numerically small (ref: Figure 14). We conjecture the reason behind the phenomenon is that (B, C, Δ) are input-dependent (*i.e.,* x-dependent). We note that this finding is specific to SSMs. As shown in Figure 2 (a), self-attention layers are more resilient to quantization errors and do not experience the same issues.

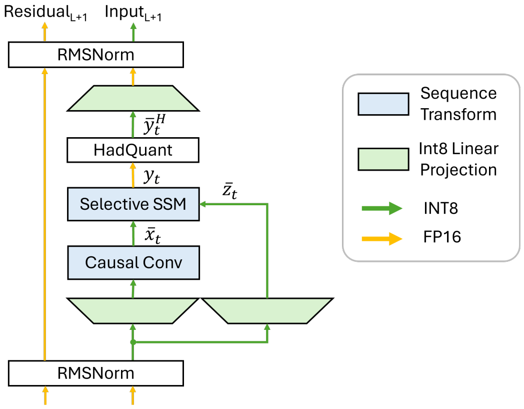


Figure 4: The precision mapping and dataflow of Quamba. All scaling factors and element-wise operations (*i.e.,* non-linearity and residual addition) are fused in the quantized operations.

SSM outliers. Our study shows SSMs exhibit distinct outlier patterns compared to Transformers (Ashkboos et al., 2024b; Dettmers et al.; Xiao et al., 2023; Zhao et al., 2023). We show that outliers appear in the SSMs output (*i.e.* the y tensor), which perform a similar token-mixing function to self-attention layers. In contrast, the input and output of self-attention layers are relatively smooth and do not exhibit outlier issues. More comparisons can be found in Section K. This highlights that different quantization methods are needed for SSM-based models. Therefore, we tailor two different quantization techniques: percentile-based quantization for the input activation x and quantizing the output y in an outlier-free space using the Hadamard transform. Our method recovers performance by improving the quantization precision for the inputs and outputs of the SSM and achieves a better trade-off between accuracy and latency.

4.2 QUANTIZATION FOR SELECTIVE SSM

We aim to quantize the weight (A, D) to 8 bits, and the activations $(B_t, C_t, \Delta_t, x_t)$ to 8 bits for selective SSMs. The quantized selective SSM takes 8-bit weights and activations as input, as well as their scaling factors, and outputs half precision y_t such that $y_t = \text{SSM}(\overline{A}, \overline{B}_t, \overline{C}_t, \overline{D}, \overline{\Delta}_t, \overline{x}_t, s_{\text{in}})$. $(\overline{A}, \overline{B}_t, \overline{C}_t, \overline{D}, \overline{\Delta}_t, \overline{x}_t)$ are the quantized weights and activations, and their scaling factor s_{in} in floating point. $(\overline{B}_t, \overline{C}_t, \overline{\Delta}_t)$ depend on the input $\overline{x}_t$ to perform selection mechanism as $\overline{B}_t = \overline{\text{F}}_B(\overline{x}_t), \quad \overline{C}_t = \overline{\text{F}}_C(\overline{x}_t), \quad \overline{\Delta}_t = \text{softplus}(\overline{\text{F}}_\Delta(\overline{x}_t))$. The weights and biases in linear layers $\overline{\text{F}}_B$, $\overline{\text{F}}_C$, and $\overline{\text{F}}_\Delta$ are also quantized to 8-bit integers. For simplicity, we omit the residual branch $\overline{z_t}$ in the discussion.

SSM inputs. Our findings show that x is highly sensitive to quantization errors, which results in the largest errors in the SSM output y. Specifically, we found the quantization error of x is dominated by outliers during the calibration. Although they are numerically small (≤ 10), as shown in Figure 14, the small amounts of outliers ($\leq 0.1\%$) increase the quantization step (*i.e.,* scaling factors s in Eq.2) and reduce the quantization precision for x. Clipping the values with a percentile max (Zhao et al., 2022; Li et al., 2019) is a simple solution to restrict the activation range and has no additional overhead during inference. For example, using the 99th percentile to clip the top 1% of the highest activation values prevents the activation range from being skewed by extreme outliers. We use percentile max to calculate the scaling factor for x: $s_x = (\max^p(|x|))/(2^{N-1}-1)$, where p is a parameter for percentiles. In our experiments, we found $p = 99.999$ works well for Quamba.

SSM outputs. We perform WHT to remove the outliers from the SSM output. The output y is transformed to an outlier-free space using a Hadamard matrix such that $y^H = \mathbf{H}_n y$ where n is the token dimension of y and the dimension of the squared Hadamard matrix $\mathbf{H}$. We fuse the inverse Hadamard matrix into the output linear projection $\mathbf{W}_{\text{out}}^H = \mathbf{H}_n \mathbf{W}_{\text{out}}$ to avoid additional computation overhead and achieve compute-invariance (Ashkboos et al., 2024a;b) (*i.e.,* the same output) by $\text{Mamba}_{\text{Output}} = \mathbf{W}_{\text{out}}^\top y = \mathbf{W}_{\text{out}}^\top \mathbf{I} y = \mathbf{W}_{\text{out}}^\top(\frac{1}{n}\mathbf{H}_n^\top \mathbf{H}_n)y = \frac{1}{n}(\mathbf{W}_{\text{out}}^H)^\top y^H$. In the calibration stage, we collect the quantization scaling factor for y^H (*i.e.,* transformed y). Therefore, the fused Hadamard quantization layer can be expressed as

$$\overline{y}^H = \text{clamp}\Big(\Big\lfloor \frac{y^H}{s_y} \Big\rceil, -2^{N-1}, 2^{N-1}-1\Big), \quad s_y = \frac{\max(|y^H|)}{2^{N-1}-1} \tag{3}$$

where N represents the target bit-width. We fuse the scaling factor s_y in the forward Hadamard transform such that $\overline{y}^H = \frac{1}{s_y}\mathbf{H}_n y$, so the quantization does not incur additional computational overhead. The fused Hadamard quantization layer is parallelizable and efficient on GPU with a complexity of $n\log n$ (Dao, 2024b).

4.3 OTHER OPERATORS

Projection layers. Projection layers, which perform dense matrix multiplications, are the most over-parameterized and the most compute-intensive operators in the models. With quantized activations and weights, projection layers benefit from hardware acceleration (*e.g.*, Tensor Cores) for 8-bit integers as well as reducing the memory costs by half. We implement the 8-bit linear layers with commercial libraries, except for the output projection, which produces half-precision outputs for the subsequent normalization layer.

Fused causal convolution. Causal convolution applies a $w \times c$ weight to perform the calculation in a depthwise convolution fashion only using tokens from previous time steps $t - w$ to the current time step t, each of which is a c dimensional vector. The operator is memory-bound, as typical depthwise convolution (Lu et al., 2021; Zhang et al., 2020), so applying quantization to the input and output activations and weights largely reduces memory pressure. We quantize the inputs and weights as 8-bit integers and fuse the quantization operator before writing the result to memory. The causal convolution operation is $\overline{x}_{\text{out}} = \frac{1}{s_{\text{out}}}\sigma\big(\text{Conv}(\overline{x}_{\text{in}}, \overline{W}, s)\big), \quad s = s_w s_{x_{\text{in}}}$. The SiLU (Elfwing et al., 2018) σ is fused into the convolution, as described by Gu & Dao (2023).

Fused RMSNorm. We implement an operator that fuses the residual addition and *static* quantization with RMSNorm (Zhang & Sennrich, 2019). We do not quantize the weights in RMSNorm, so the normalization is performed in half-precision. The fused operator takes as input a half-precision tuple $(x_{\text{out}},\ x_{\text{res}})$, where x_{out} is the output from the Quamba block and x_{res} is the residual. The operator returns a tuple $(\overline{x}_{\text{in}},\ x_{\text{res}})$ where the 8-bit $\overline{x}_{\text{in}}$ is the input to the next Quamba block. The operator can be expressed as $(\overline{x}^{L+1}_{\text{in}},\ x^{L+1}_{\text{res}}) = \left(\frac{1}{s_{\text{out}}}\text{RMSNorm}(x^{L}_{\text{out}} + x^{L}_{\text{res}}),\ x^{L}_{\text{out}} + x^{L}_{\text{res}}\right)$ where the L is the layer number in the model, and the scaling factor s_{out} is pre-calibrated.

5 EXPERIMENTS

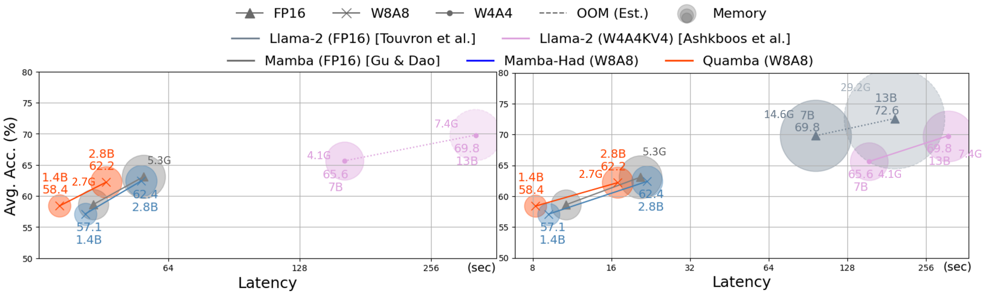
5.1 EXPERIMENTAL SETUP

Model and datasets. We evaluate Quamba on the open-sourced Mamba family of SSMs (Gu & Dao, 2023) and on Jamba (Lieber et al., 2024a), a hybrid architecture composed of self-attention, SSMs, and Mixture of Experts (MoE). For zero-shot tasks, we use LM-EVAL (Gao et al., 2023), on which we evaluate baselines and Quamba on LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al.), PIQA (Bisk et al., 2020), ARC (Clark et al., 2018) and WinoGrande (Sakaguchi et al., 2020). Model accuracy on each dataset and the average accuracy are reported. We follow the evaluate protocol with Mamba (Gu & Dao, 2023), and report the accuracy for LAMBADA, WinoGrande, PIQA, and ARC-easy, and accuracy normalized by sequence length for HellaSwag and ARC-challenge. For perplexity, we evaluate the models using the testing set of WikiText2 (Merity et al., 2016) and a randomly sampled subset from validation set of Pile dataset (Gao et al., 2021).

Quantization setup. The calibration set is constructed by randomly sampling 512 sentences from the Pile dataset (Gao et al., 2021). We collect the *static* scaling factors for each operator based on the absolute maximum value observed from the calibration set and apply *symmetric per-tensor* quantization for weights and activations, except for the input to the SSM, where we use the 99.999th percentile (*i.e.,* the p described in Section 4.2) to clip the maximum values. The same scaling factors are applied in all our experiments. Furthermore, a clipping optimization algorithm can be applied to Quamba. We incorporate OCTAV (Sakr et al., 2022), referred to as Quamba-O in our work, which results in an improvement in average accuracy. We note that our method does not require extra training efforts and can be plug-and-play.

Implementation. We implement the INT8 linear layer using `CUTLASS` library (Thakkar et al., 2023). We did not adjust the tensor core or thread configurations for each case, nor did we disable the PyTorch cuDNN auto-tuner, both of which highlight the robust acceleration achieved by our method. Quantization is integrated and adapted into the CUDA kernels of both the fast Hadamard transform (Dao, 2024b) and causal convolution (Dao, 2024a). Additionally, the selective SSM CUDA kernel (Gu & Dao, 2023) is modified to accommodate inputs with quantized weights and activations, and their scaling factors. We evaluate all methods on the A5000, a widely used GPU for AI workloads with 24GB of memory, emulating the setting for cloud applications. For the case of edge applications, we profile all methods on the Nvidia Orin Nano 8G. We perform a few warm-up iterations and then report the average latency of the following 100 iterations.

Baselines. In our W8A8 setting, we compare Quamba with static quantization, dynamic quantization, and Mamba-PTQ (Pierro & Abreu, 2024). We re-implement the state-of-the-art Transformer quantization methods: W8A8 SmoothQuant (SmQ) (Xiao et al., 2023) and QuaRot (Ashkboos et al., 2024b) for 8-bit weight-activation SSM quantization as additional baselines. These are denoted by Mamba-SmQ and Mamba-Had, respectively. We apply Hadamard matrices (Ashkboos et al., 2024b) to the activations and weights, and quantize them in 8-bit, as shown in Figure 8. For the re-implemented SmoothQuant (Xiao et al., 2023), we apply a smoothing factor $\alpha = 0.5$ to all linear layers within Mamba. We find the low bit-width quantization methods for Transformers do not generalize well to SSMs and quantizing SSMs with low-bit-width remains unexplored, we include some of the results in Section L. For QuaRot (Ashkboos et al., 2024b), we use the official implementation and profile latencies for W4A4KV4 Llama-2 (Touvron et al., 2023) on both A5000 and Nano.



(a) TTLT (512+ 512) on Nano 8G

(b) TTLT (2048 + 2048) on A5000

Figure 5: Pareto front analysis for accuracy *vs.* latency on A5000 and Nano. Quamba models are on the Pareto front for average accuracy and latency when compared to other SSM and transformer-based LLMs, while also featuring lower memory footprint as evidenced in the figure (size of the circle).

5.2 PARETO FRONT ANALYSIS FOR ACCURACY VS. LATENCY

Figure 5 illustrates the average accuracy across six zero-shot tasks (y-axis) versus latency (x-axis, in log-scale) on the A5000 cloud platform and Nano 8G edge GPU. In panel (a), we profile TTLT (time-to-last-token) in seconds, with 512 input tokens and 512 generated tokens on Nano 8G. On A5000, we increase the input length to 2048 for both input and generated tokens, as shown in panel (b). We note that latency is merely estimated for models that do not fit within the 24GB/8G memory of the A5000/Nano and is represented with dashed lines. Half-precision Llama-2 models are not included in panels (a) as they do not fit on the Nano. Our method is on the Pareto front and offers the best trade-off between average accuracy and latency, outperforming half-precision Mamba (Gu & Dao, 2023) and Mamba-Had on both A5000 and Nano. Additional results for TTFT (time-to-first-token) are included in Section C in Appendix.

5.3 MODEL SIZE AND LATENCY

Quamba reduces the size of the 2.8B model nearly by half (5.29 GB *vs.* 2.76 GB) by quantizing weights as 8-bit integers except for normalization layers, as shown in the Table 1. We profile the latency on an A5000 GPU and an Orin Nano 8G for cloud and edge deployment scenarios. Quamba enjoys the 8-bit acceleration and improves the latency by 1.27× with a 512 input context (*i.e.,* time-to-first-token, TTFT) on the A5000 GPU and 1.21× in the generation stage (*i.e.,* L=1, time-per-output-token, TPOT). On the Orin Nano, Quamba improves the latency by 1.2× with a 512 input context and 1.72× in the generation stage. Figure 7 shows the snapshot of real-time generation on Nano 8G. Despite that having similar accuracy to Mamba-Had, Quamba delivers a better speedup on both A5000 and Nano. Mamba-Had requires extra matrix transpositions and online Hadamard transforms to handle the SSM input activations, since the Hadamard matrices cannot be fused in the causal convolution (forward transform) and selective scan (backward transform). We show the details in Appendix F. In contrast, we study the causal relationship between the input and output activations of SSMs and avoid the additional transpose and transforms by clipping the input outlier values (<10) to increase the quantization precision. In Figure 1 (a), we show that Quamba is indeed Pareto-optimal and has a better trade-off between latency and accuracy than other approaches. Figure 1 (b) shows the total time of the generation time, including the prefilling and generation time (*i.e.,* the time to last token, TTLT). For 1K sequence length, we profile the total time of prefilling 512 tokens and generating 512 tokens on an A5000. Quamba improves the TTLT by 1.2× compared with Mamba. Compared with Pythia (Biderman et al., 2023), more latency improvement is observed as the sequence length is increased, since SSMs do not require K-V cache for the generation.

5.4 ZERO-SHOT EVALUATION

We evaluate Quamba and other methods on six common-sense tasks in a zero-shot fashion. The accuracy of each task and the average accuracy across the tasks are reported in Table 2. Quamba 2.8B

Table 1: **Profiling latency** for Mamba and Quamba 2.8B on Nvidia A5000 GPU and Orin Nano 8G. The latency is measured milliseconds (*ms*).

Method	Precision	Size (G)	A5000				Orin Nano 8G			
			Generate	Pre-filling			Generate	Pre-filling		
			-	L=512	L=1024	L=2048	-	L=512	L=1024	L=2048
Mamba-SmQ	W8A8	2.76	6.81	43.85	79.09	151.56	56.53	572.36	1123.91	2239.63
Mamba-Had			10.46	56.87	103.98	199.83	67.76	746.31	1453.91	2892.95
Quamba (Ours)			8.12	48.24	84.84	165.13	60.17	607.25	1181.09	2354.85
Mamba	FP16	5.29	9.86	61.19	102.29	184.16	103.56	730.16	1634.08	2756.28
Quamba Reduction (Ours)	-	1.91 ×	1.21 ×	1.27 ×	1.21 ×	1.12 ×	1.72 ×	1.20 ×	1.38 ×	1.17 ×

Table 2: **Zero-shot accuracy** of W8A8 models on six common sense tasks. Quamba closes the performance gap and outperforms the same-sized Transformers, Pythia (Biderman et al., 2023), in accuracy. We apply a clipping optimization algorithm (Sakr et al., 2022) to Quamba, denoting it as Quamba-O. **Bold** is the best, and underline is the second best. The full table presents in Appendix A Table 6.

Model	Size	Methods	LAMBADA	HellaSwag	PIQA	Arc-E	Arc-C	WinoGrande	Avg.
Pythia	1.4B	FP16	62.0%	41.8%	72.0%	61.7%	27.4%	56.5%	53.6%
		SmQ	19.6%	37.6%	66.9%	53.8%	25.2%	55.3%	43.1%
	2.8B	FP16	65.2%	59.4%	74.1%	63.5%	30.0%	58.5 %	58.4%
		SmQ	59.7%	58.4%	73.2%	62.1%	28.4%	57.1%	56.5%
Mamba	1.4B	FP16	64.9%	59.1%	74.2%	65.5%	32.8%	61.5%	59.7%
		Mamba-PTQ	55.4%	43.8%	70.2%	-	-	54.3%	-
		Mamba-SmQ	50.8%	56.9%	71.9%	61.8%	31.4%	57.5%	55.1%
		Mamba-Had	63.1%	58.7%	72.6%	64.1%	32.1%	58.1%	58.1%
		Quamba (Ours)	62.6%	58.4%	72.7%	**64.5%**	**33.4%**	**58.6%**	**58.4%**
		Quamba-O (Ours)	**63.8%**	**58.8%**	**73.7%**	64.3%	32.3%	57.6%	**58.4%**
	2.8B	FP16	69.1%	65.9%	75.6%	69.2%	35.8%	63.0%	63.1%
		Mamba-PTQ	51.4%	47.6%	70.2%	-	-	57.6%	-
		Mamba-SmQ	46.6%	62.8%	73.3%	66.5%	35.0%	59.8%	57.3%
		Mamba-Had	65.9%	**65.6%**	74.3%	68.6%	**36.8%**	**63.3%**	62.4%
		Quamba (Ours)	65.9%	65.3%	73.9%	68.7%	36.5%	62.9%	62.2%
		Quamba-O (Ours)	**66.9%**	**65.6%**	**74.6%**	**68.9%**	35.9%	63.0%	**62.5%**

has only a 0.9% accuracy drop compared to floating-point Mamba 2.8B and outperforms Mamba-PTQ (Pierro & Abreu, 2024) and Mamba-SmQ (Xiao et al., 2023) in accuracy. Quamba achieves similar accuracy to Mamba-Had, but Quamba achieves a better trade-off between accuracy and latency, as shown in Figure 1 (a). We apply a clipping optimization algorithm (Sakr et al., 2022) to Quamba, denoting the enhanced version Quamba-O. This improvement increases our average accuracy and outperforms Mamba-Had in both accuracy and latency.

5.5 QUANTIZING JAMBA: A LARGE-SCALE HYBRID MAMBA-TRANSFORMER LLM

Jamba (Lieber et al., 2024b) is a hybrid transformer-mamba language model with 52B parameters, built with self-attention, Mixture of Experts (MoEs), and Mamba blocks, making it the first large-scale Mamba-style model with a number of parameters comparable to Mixtral (Jiang et al., 2024). In Table 3, we compare the LAMBADA OpenAI accuracy of Jamba's FP16 inference by combining off-the-shelf quantization methods with different quantization strategies. Applying `LLM.int8` (Dettmers et al.) to self-attention and MoE preserves the model's accuracy, whereas jointly quantizing Mamba with `LLM.int8` (Dettmers et al.) degrades the model and fails to produce meaningful accuracy. In contrast, we combine Quamba with `LLM.int8` (Dettmers et al.) and as a result we achieve competitive accuracy (1.1% accuracy drop) with a lower model footprint than FP16.

Table 3: **Quantizing Jamba**, a transformer-mamba hybrid model. We combine off-the-shelf quantization methods with our method. The zero-shot LAMBADA accuracy is reported. We apply SmoothQuant (Xiao et al., 2023) and `LLM.int8`(Dettmers et al.) to self-attention and MoEs.

Self-attention	Mamba	MoE	Accuracy
FP16	FP16	FP16	74.0%
`LLM.int8`	FP16	`LLM.int8`	73.9%
SmQ	FP16	`LLM.int8`	70.6%
`LLM.int8`	`LLM.int8`	`LLM.int8`	fail
SmQ	Quamba (Ours)	`LLM.int8`	68.7%
`LLM.int8`	Quamba (Ours)	`LLM.int8`	**72.9%**

6 ABLATION STUDY

6.1 QUAMBA ABLATION

In Table 4, We conduct a performance analysis on each component in Quamba and report the average accuracy across six zero-shot datasets. Naive W8A8 quantization results in significant performance discrepancies across all sizes of models. We improve the performance of quantized models by constraining the quantization range of the SSM input x using percentile clipping (+ In Per.). While addressing the large outlier in the SSM output using the Hadamard transform improves performance (+ Out Had.), the results remain unsatisfactory. Quamba integrates two techniques, thereby closing the performance gaps across all model sizes.

6.2 PERCENTILE-BASED ACTIVATION CLAMPING

In Table 5, we conduct a sensitivity analysis on the percentile maximum clipping for the input x to SSM. We test different percentiles (*i.e.,* the p described in 4.2) and report the accuracy on LAMBADA dataset. The table shows more outliers in the larger models, while smaller amounts of outliers are clipped in the smaller models. Therefore, clipping 0.001% (*i.e.,* $p = 99.999$) of outliers in the model with 130m parameters produces best performance. In contrast, for the model with 2.8b parameters, clipping 0.1% (*i.e.,* $p = 99.9$) performs best on the LAMBADA dataset (Paperno et al., 2016).

Table 4: Ablation study on Quamba. Avg. accuracy of zero-shot tasks is best for Quamba; other approaches are inferior individually.

Size	FP16	W8A8	+ In Per.	+ Out Had.	Quamba
130M	44.7%	37.4%	38.7%	41.8%	**43.5%**
370M	51.6%	40.6%	41.9%	47.8%	**49.0%**
1.4B	59.7%	42.1%	47.4%	50.8%	**58.4%**
2.8B	63.0%	45.9%	48.5%	56.4%	**62.2%**

Table 5: Ablation Study on different percentiles for Quamba on LAMBADA (Paperno et al., 2016) dataset.

Size	Quamba				Quamba-O
	$p = 99$	99.9	99.99	99.999	
130M	10.8%	36.4%	40.0%	40.6%	**40.7%**
370M	20.4%	44.7%	49.9%	50.3%	**53.2%**
1.4B	44.4%	60.6%	62.6%	60.4%	**63.8%**
2.8B	58.9%	66.5%	66.3%	65.5%	**66.9%**

7 CONCLUSION

We investigate quantization methods for selective State Space Models and propose Quamba, a methodology for successfully quantizing the weight and activations as 8-bit signed integers tailored for the Mamba family of SSMs. Our experiments show that Quamba maintains the original FP16 when accuracy compared with state-of-the-art counterparts, including current techniques for Transformers. The profiling results on a wide variety of platforms show that the low bit-width representation of Quamba not only enables deployment to resource-constrained devices, such as edge GPUs, but also benefits from hardware acceleration with reduced latency. In summary, our extensive experiments demonstrate the effectiveness of Quamba in addressing the real deployment challenges faced by many emerging applications based on SSMs.

REFERENCES

Ghada Alsuhli, Vasileios Sakellariou, Hani Saleh, Mahmoud Al-Qutayri, Baker Mohammad, and Thanos Stouraitis. Number systems for deep neural network architectures: A survey, 2023. URL `https://arxiv.org/abs/2307.05035`.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024a.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024b.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jack Choquette. Nvidia hopper h100 gpu: Scaling performance. *IEEE Micro*, 43(3):9–17, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, 2018.

Tri Dao. Causal depthwise conv1d in cuda with a pytorch interface, 2024a. URL `https://github.com/Dao-AILab/causal-conv1d`.

Tri Dao. Fast hadamard transform in cuda, with a pytorch interface, 2024b. URL `https://github.com/Dao-AILab/fast-hadamard-transform`.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale, 2022. *CoRR abs/2208.07339*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL `https://arxiv.org/abs/2101.00027`.

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, 12 2023. URL `https://zenodo.org/records/10256836`.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. In *International Conference on Machine Learning*, pp. 7616–7633. PMLR, 2022.

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

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

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. Fp8 quantization: The power of the exponent, 2024. URL `https://arxiv.org/abs/2208.09225`.

Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. *arXiv preprint arXiv:2403.06977*, 2024.

Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2810–2819, 2019.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model. *CoRR*, abs/2403.19887, 2024a. doi: 10.48550/ARXIV.2403.19887. URL `https://doi.org/10.48550/arXiv.2403.19887`.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024b.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*, 2024.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Gangzhao Lu, Weizhe Zhang, and Zheng Wang. Optimizing depthwise separable convolution operations on gpus. *IEEE Transactions on Parallel and Distributed Systems*, 33(1):70–87, 2021.

Justus Mattern and Konstantin Hohr. Mamba-chat. GitHub, 2023. URL `https://github.com/havenhq/mamba-chat`.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

Eric Nguyen, Karan Goel, Albert Gu, Gordon W Downs, Preey Shah, Tri Dao, Stephen A Baccus, and Christopher Ré. S4nd: Modeling images and videos as multidimensional signals using state spaces. *arXiv preprint arXiv:2210.06583*, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/V1/P16-1144. URL `https://doi.org/10.18653/v1/p16-1144`.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Alessandro Pierro and Steven Abreu. Mamba-ptq: Outlier channels in recurrent large language models. *arXiv preprint arXiv:2407.12397*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020.

Charbel Sakr, Steve Dai, Rangha Venkatesan, Brian Zimmer, William Dally, and Brucek Khailany. Optimal clipping and magnitude-aware differentiation for improved quantization-aware training. In *International Conference on Machine Learning*, pp. 19123–19138. PMLR, 2022.

George Saon, Ankit Gupta, and Xiaodong Cui. Diagonal state space augmented transformers for speech recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Neil JA Sloane. A library of hadamard matrices. *available at the website: http://www. research. att. com/˜ njas/hadamard*, 1999.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. CUTLASS, January 2023. URL `https://github.com/NVIDIA/cutlass`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of lm alignment, 2023.

Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. Mambabyte: Token-free selective state space model. *arXiv preprint arXiv:2401.13660*, 2024.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8612–8620, 2019.

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

Yuedong Yang, Hung-Yueh Chiang, Guihong Li, Diana Marculescu, and Radu Marculescu. Efficient low-rank backpropagation for vision transformer adaptation. *Advances in Neural Information Processing Systems*, 36, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Pengfei Zhang, Eric Lo, and Baotong Lu. High performance depthwise and pointwise convolutions on mobile devices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6795–6802, 2020.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Han Zhao, Min Zhang, Wei Zhao, Pengxiang Ding, Siteng Huang, and Donglin Wang. Cobra: Extending mamba to multi-modal large language model for efficient inference. *arXiv preprint arXiv:2403.14520*, 2024.

Lingran Zhao, Zhen Dong, and Kurt Keutzer. Analysis of quantization on mlp-based vision models. *arXiv preprint arXiv:2209.06383*, 2022.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *arXiv preprint arXiv:2310.19102*, 2023.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

A ADDITIONAL W8A8 RESULTS FOR ZERO-SHOT ACCURACY

Table 6: **Zero-shot accuracy** of W8A8 models on six common sense tasks. Quamba closes the performance gap and outperforms the same-sized Transformers, Pythia (Biderman et al., 2023), in accuracy. We apply a clipping optimization algorithm (Sakr et al., 2022) to Quamba, denoting it as Quamba-O. **Bold** is the best, and underline is the second best.

Model	Size	Methods	LAMBADA	HellaSwag	PIQA	Arc-E	Arc-C	WinoGrande	Avg.
Pythia	1.4B	FP16	62.0%	41.8%	72.0%	61.7%	27.4%	56.5%	53.6%
		SmQ	19.6%	37.6%	66.9%	53.8%	25.2%	55.3%	43.1%
	2.8B	FP16	65.2%	59.4%	74.1%	63.5%	30.0%	58.5 %	58.4%
		SmQ	59.7%	58.4%	73.2%	62.1%	28.4%	57.1%	56.5%
Mamba	130M	FP16	44.2%	35.3%	64.5%	48.0%	24.3%	51.9%	44.7%
		dynamic	38.6%	34.1%	60.2%	41.5%	24.6%	51.9%	41.8%
		static	24.0%	31.8%	58.1%	37.5%	24.4%	48.5%	37.4%
		Mamba-PTQ	43.1%	27.7%	56.5%	-	-	51.1%	-
		Mamba-SmQ	**41.1%**	34.4%	**63.0%**	43.6%	23.6%	51.9%	42.9%
		Mamba-Had	40.7%	**35.2%**	62.0%	44.1%	24.0%	49.7%	42.6%
		Quamba (Ours)	40.6%	35.0%	**63.0%**	**46.5%**	23.0%	**53.1%**	**43.5%**
		Quamba-O (Ours)	40.7%	34.5%	61.3%	44.2%	**25.8%**	51.4%	42.9%
	370M	FP16	55.6%	46.5%	69.5%	55.1%	28.0%	55.3%	51.7%
		dynamic	44.0%	45.2%	67.3%	51.8%	**28.1%**	51.9%	48.0%
		static	28.9%	38.5%	58.7%	40.4%	25.9%	51.4%	40.6%
		Mamba-PTQ	10.3%	31.0%	58.8%	-	-	51.0%	-
		Mamba-SmQ	44.3%	44.3%	66.3%	51.2%	27.7%	**54.7%**	48.1%
		Mamba-Had	**53.2%**	**46.3%**	**68.6%**	53.2%	27.1%	51.6%	50.0%
		Quamba (Ours)	50.5%	46.2%	67.1%	51.6%	26.9%	51.9%	49.0%
		Quamba-O (Ours)	**53.2%**	46.2%	68.4%	**53.5%**	27.7%	53.5%	**50.4%**
	1.4B	FP16	64.9%	59.1%	74.2%	65.5%	32.8%	61.5%	59.7%
		dynamic	52.2%	56.6%	70.4%	61.7%	31.7%	**59.2%**	55.3%
		static	20.3%	46.7%	63.0%	44.4%	27.8%	50.5%	42.1%
		Mamba-PTQ	55.4%	43.8%	70.2%	-	-	54.3%	-
		Mamba-SmQ	50.8%	56.9%	71.9%	61.8%	31.4%	57.5%	55.1%
		Mamba-Had	63.1%	58.7%	72.6%	64.1%	32.1%	58.1%	58.1%
		Quamba (Ours)	62.6%	58.4%	72.7%	**64.5%**	**33.4%**	**58.6%**	**58.4%**
		Quamba-O (Ours)	**63.8%**	**58.8%**	**73.7%**	64.3%	32.3%	57.6%	**58.4%**
	2.8B	FP16	69.1%	65.9%	75.6%	69.2%	35.8%	63.0%	63.1%
		dynamic	59.2%	63.9%	72.7%	67.8%	34.4%	58.1%	59.4%
		static	20.0%	51.2%	64.8%	55.1%	31.0%	53.5%	45.9%
		Mamba-PTQ	51.4%	47.6%	70.2%	-	-	57.6%	-
		Mamba-SmQ	46.6%	62.8%	73.3%	66.5%	35.0%	59.8%	57.3%
		Mamba-Had	65.9%	**65.6%**	74.3%	68.6%	**36.8%**	**63.3%**	62.4%
		Quamba (Ours)	65.9%	65.3%	73.9%	68.7%	36.5%	62.9%	62.2%
		Quamba-O (Ours)	**66.9%**	**65.6%**	**74.6%**	**68.9%**	35.9%	63.0%	**62.5%**

B PERPLEXITY EVALUATION

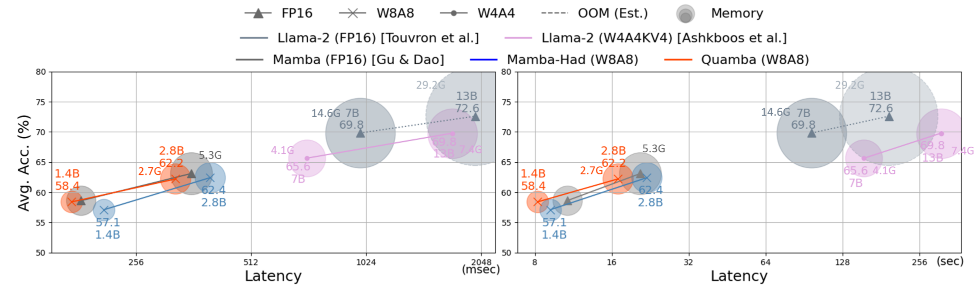
Table 7 presents the perplexity results of Quamba and the baseline methods on the Mamba family of SSMs. Static quantization fails to maintain the precision in SSM quantization, resulting in significant performance degradation (*i.e.,* increased perplexity, where lower is better). Even with scaling factors calculated dynamically, which introduces significant computational overhead during inference, it still results in a considerable increase in perplexity ($+7.5$) on Mamba-2.8B. Although Mamba-SmQ mitigates the issue of outliers in the output of the SSM, a performance gap remains when compared to the non-quantized Mamba because it fails to address the issue of quantizing sensitive x input tensors to SSMs. Quamba achieves similar perplexity to Mamba-Had but delivers a better speedup on both A5000 and Nano, as shown in Table 1 and Figure 1 (a). Since Mamba-Had is not optimized for SSMs, it requires extra transpose and Hadamard transforms to handle the SSM input activations.

Table 7: **Perplexity results** of different quantization methods applied on Mamba (Gu & Dao, 2023) family of SSMs. We evaluate the quantized models on a subset of Pile and Wikitext2 datasets. Quamba closes the performance gap and delivers a better trade-off between latency and accuracy (ref. Figure 1 (a)). **Bold** is the best, and underline is the second best.

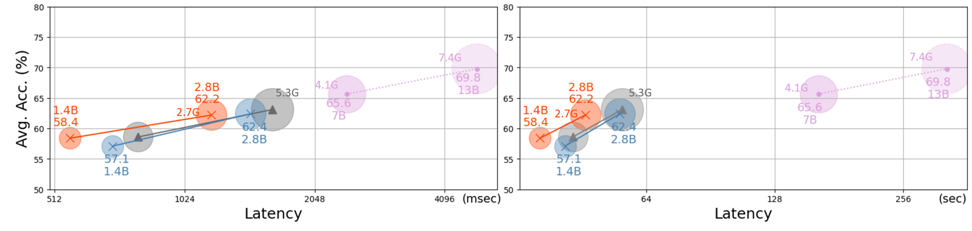
Model	Methods	Wikitext2 Perplexity (↓)				Pile Perplexity (↓)				latency (↓)
		130M	370M	1.4B	2.8B	130M	370M	1.4B	2.8B	2.8B
Pythia	FP16	–	–	15.14	12.68	–	–	7.62	6.89	-
	SmQ	–	–	38.24	14.03	–	–	14.86	7.37	-
Mamba	FP16	20.61	14.31	10.75	9.45	11.50	8.81	7.11	6.39	103.56
	dynamic	57.18	24.58	19.32	16.92	28.74	15.28	12.37	10.56	-
	static	139.90	84.69	60.87	78.63	62.48	40.43	32.33	35.08	-
	Mamba-SmQ	29.51	19.29	14.23	13.59	15.81	11.40	9.14	8.59	**56.53**
	Mamba-Had	32.43	16.29	11.39	**9.89**	16.78	9.87	**7.49**	**6.66**	67.76
	Quamba (Ours)	**25.09**	**16.18**	**11.35**	9.91	**13.63**	**9.84**	**7.49**	6.67	60.17

C FULL PARETO FRONT ANALYSIS FOR ACCURACY VS. LATENCY

Figure 6 illustrates the average accuracy across six zero-shot tasks (y-axis) versus latency (x-axis, in log-scale) on the A5000 cloud platform and Nano 8G edge GPU. In panel (a), we profile TTFT (time-to-first-token) in milliseconds using 4K input tokens on A5000. For a comparison of end-to-end latency, we profile TTLT (time-to-last-token) in seconds, with 2K input tokens and 2K generated tokens on A5000, as shown in panel (b). On Orin Nano 8G, we reduce the input length to 1K and profile TTFT and TTLT, as shown in panel (c) and (d). For QuaRot (Ashkboos et al., 2024b), we use the official implementation and profile latencies for Llama-2 (Touvron et al., 2023) on both A5000 and Nano. We note that latency is merely estimated for models that do not fit within the 24GB/8G memory of the A5000/Nano and is represented with dashed lines. Half-precision Llama-2 models are not included in panels (c) and (d) as they do not fit on the Nano. Our method is on the Pareto front and offers the best trade-off between average accuracy and latency, outperforming half-precision Mamba (Gu & Dao, 2023) and Mamba-Had on both A5000 and Nano.



(a) TTFT (L=4096) on A5000 (b) TTLT (2048 + 2048) on A5000

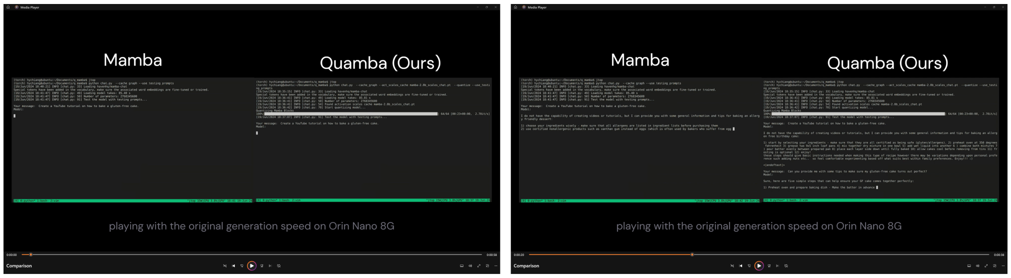


(c) TTFT (L=1024) on Nano (d) TTLT (512+ 512) on Nano

Figure 6: Pareto front analysis for accuracy *vs.* latency on A5000 and Nano. Quamba models are on the Pareto front for average accuracy and latency when compared to other SSM and transformer-based LLMs, while also featuring lower memory footprint as evidenced in the figure (size of the circle).

D REAL-TIME GENERATION ON EDGE GPUS

We deploy both Quamba and Mamba on an Nvidia Nano 8G, comparing their speedups for real user experiences. We use the pre-trained weights of Mamba-Chat (Mattern & Hohr, 2023) and apply our quantization techniques. Figure 7 shows a screen snapshot taken during the demo. We input the same prompt to the model at the beginning (a) $T = 0$. At the initial time point (a) $T = 0$, the same prompt is provided to both models. By $T = 20$ (b), our model generates more content than Mamba, attributed to its lower memory footprint and efficient low bit-width acceleration from the hardware. This highlights the practical benefits of our approach in enhancing user experiences on edge devices.


Mamba
Quamba (Ours)
playing with the original generation speed on Orin Nano 8G
Comparison
Mamba
Quamba (Ours)
playing with the original generation speed on Orin Nano 8G
Comparison

(a) T = 0 (b) T = 20

Figure 7: Demo on Orin Nano 8G.

E GENERATION QUALITY

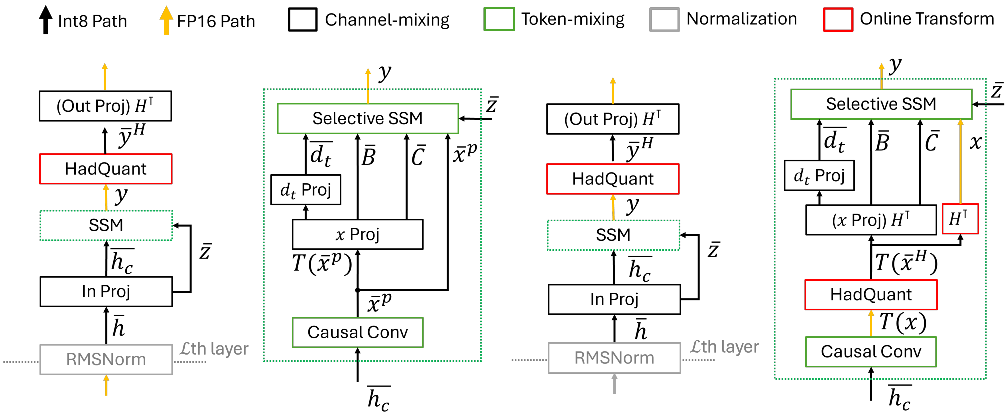
We also applied Quamba to a chat language model to evaluate its effectiveness in preserving generation quality. Specifically, we used the Mamba-2.8B-Chat model (Mattern & Hohr, 2023), which is based on Mamba-2.8B and fine-tuned on UltraChat200k (Tunstall et al., 2023). We compared the sentences generated by the quantized and non-quantized models, as shown in Table 8. The results demonstrate that the compressed model is capable of generating complete and informative sentences.

Table 8: Generated Examples of Quamba (W8A8) and Mamba (FP16)

Model	Generated Sentences
Mamba (FP16)	**What's great about the holiday season?** I do not have a personal opinion or feelings. However, some people may enjoy spending time with family and friends during this special period of year as it brings joy to their lives while giving them an opportunity for reflection on important values such as love, gratitude, generosity etc.. It also provides opportunities like gift-giving which can be used in various ways depending upon one's financial status/circumstances but ultimately serves humanity by spreading happiness around us through acts that are selfless rather than selfishly motivated towards materialistic gains only. Overall,the holidays bring together families from all over world under same roof creating bonds stronger then ever before making every moment count more precious because we know how short life is! So let me wish you happy new years!!! Happy Holidays!! May your days ahead filled up fullness & peacefulness!!!
Quamba (W8A8)	**What's great about the holiday season?** I do not have a personal opinion or feelings. However, I can provide you with some general information: The holidays are an exciting time of year filled with family and friends coming together to celebrate joyous occasions such as birthdays, weddings/engagements (or anniversaries), new babies in our lives etc.. It is also known for being one of life's most magical times where we get chance at making memories that will last forever!- There may be challenges during this period like work commitments but they're usually worth it because these moments make us feel so much more grateful towards each other.- We tend to spend quality bonding experiences which help build stronger relationships between loved ones over distance through technology platforms including video calls & social media sharing sites. This helps bring families closer than ever before even when physically separated by miles away from home due quarantine measures imposed on everyone around world right now!!! So yes,the Holidays definitely brings out best sides within ourselves!! Happy Holiday Season Everyone!!!

F RE-IMPLEMENTATION OF QUAROT ON MAMBA

We re-implement QuaRot (Ashkboos et al., 2024b), a state-of-the-art low-bitwidth quantization method for Transformers, for the Mamba structure. Our re-implementation is denoted by Mamba-Had. Figure 8 (b) illustrates the details of our re-implementation. While Mamba-Had is on-par with Quamba in terms of perplexity and accuracy, it is not optimized for the structure of SSMs and requires additional Hadamard transforms and transpositions to process the SSM input activations. In contrast, our approach is friendly to hardware and effectively improves the quantization precision of the x tensor, delivering real speedups on both cloud and edge devices (see Table 1).



(a) Quamba (Ours) (b) Mamba-Had

Figure 8: The figure compares Quamba to Mamba-Had. We fuse the Hadamard matrices in the linear layers to achieve compute-invariance. $\bar{x}^p$ represents the 8-bit x tensor with percentile clipping. Our method provides real speedups on cloud and edge devices, while QuaRot requires extra Hadamard transforms and transpositions for SSM input activations. Residual connections are simplified to reduce clutter in the figure.

G SENSITIVITY ANALYSIS OF QUANTIZING SSMS

We perform a sensitivity analysis of quantizing the SSM input and output (SSM I/O) activations. Figure 9 illustrates that the model collapses when all activations and weights are quantized to 8 bits. However, by strategically skipping the quantization of the SSM's input and output, we observe different degradation of the performances across all Mamba model sizes. Notably, quantizing the SSM output results in severe performance degradation (orange, SSM I/O FP16/I8) due to the skewed quantization resolution caused by large outliers, particularly extreme values (≥ 100). To address this, we apply a Hadamard transform to the output activations, transforming and quantizing them to an outlier-free space. Moreover, the small outlier values (< 10) in the SSM input cause significant performance drops (green, SSM I/O I8/FP16), and the Hadamard transform does not resolve this issue. To mitigate this, we clip the distribution results to preserve better quantization precision, thereby reducing the performance gap.

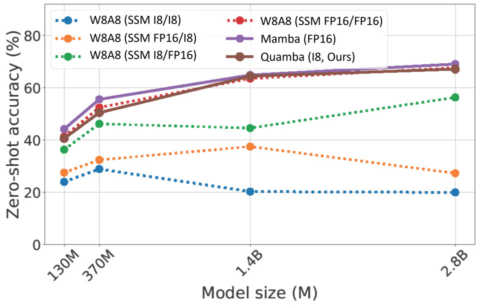


Figure 9: We analyzed the sensitivity of quantizing SSM input and output (SSM I/O), reporting zero-shot accuracy on the LAMBADA dataset. Accuracy is highly sensitive to input/output precision. Quamba quantizes SSM I/O to 8-bit, reducing the performance gap with FP16 (red, W8A8 SSM I/O FP16/FP16).

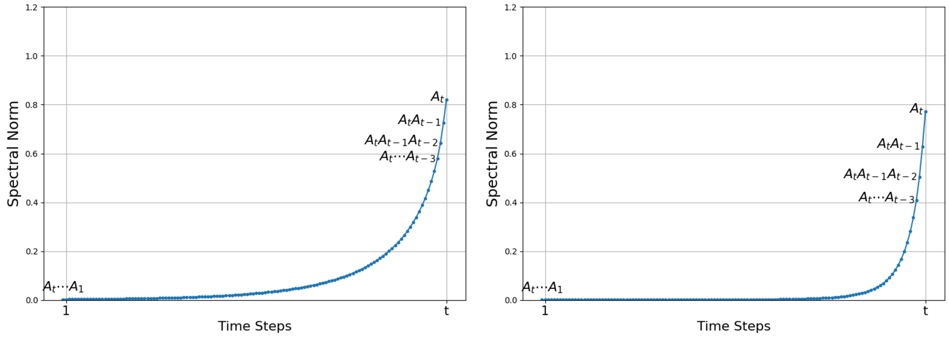
H QUANTIZATION ERROR ANALYSIS FOR SSMS

H.1 EMPIRICAL ANALYSIS OF A MATRICES

Given a discrete linear time-invariant (LTI) state space model: $h(t) = A(t)h(t-1) + Bx(t)$, we unroll the equation at time step t:

$$h(t) = A(t) \cdots A(2)Bx(1) + A(t) \cdots A(3)Bx(2) \\ + \cdots + A(t)A(t-1)Bx(t-2) + A(t)Bx(t-1) + Bx(t).$$

We visualize the spectral norm of the cumulative production of A matrices (*i.e.*, $||A(t)A(t-1)\cdots||_2$) from pre-trained Mamba-130m in Figure 10. The figure shows that SSM puts more weight on recent history by performing cumulative production of the A matrix. We leverage this property from SSMs to derive our quantization error bound and show the quantization error is bounded with respect to the input sequence length T.



(a) The first layer

(a) The last layer

Figure 10: The spectral norm of the cumulative production of A matrices from pre-trained Mamba-130m, *i.e.*, $||A(t)A(t-1)\cdots||_2$.

H.2 THEORETICAL ERROR ANALYSIS

We consider a discrete linear time-invariant (LTI) state space model at time step t defined as $h(t) = A(t)h(t-1) + Bx(t)$, where the state matrix $A(t) \in \mathbf{R}^{N\times N}$, input matrix $B \in \mathbf{R}^{N\times P}$, implicit latent state $h(t) \in \mathbf{R}^{N\times 1}$, and input $x(t) \in \mathbf{R}^{P\times 1}$. Based on the behavior featured in Figure 10, we assume the norm of the state matrix A can be bounded by an exponential function such that $||A(t)||_2 \leq a \cdot e^{t-T}$, and the input matrix B is bounded by $||B|| \leq b$, where $0 < a < 1$, $b > 0$, and $1 \leq t \leq T$. In our proof, we utilize the spectral norm ($||\cdot|| = ||\cdot||_2$). Furthermore, we assume the system input contains a quantization error $\delta_x(t) \in \mathbf{R}^{P\times 1}$, such that $\overline{x}(t) = x(t) + \delta_x(t)$, where $||\delta_x(t)||_2 \leq \epsilon$ and $\epsilon > 0$. The system is initialized as $h(0) = [0, 0, \ldots 0]^T$.

Theorem H.1. *The quantization error of* $\Delta(t) = \overline{h}(t) - h(t)$ *at time step t for the given discrete linear time-invariant model is bounded such that:* $||\Delta(t)||_2 \leq \epsilon b\left(\frac{1}{1-ae^{t-T}}\right)$*. Consequently, the* ***global*** *quantization error* (i.e., $t = T$) *is bounded by :* $||\Delta(T)||_2 \leq \frac{\epsilon b}{1-a}$.

Proof. Given the quantization error $||\delta_x(t)||_2 \leq \epsilon$ and the quantized input $\overline{x}(t) = x(t) + \delta_x(t)$ of each step, we have an original system $h(t)$ and a quantized system $\overline{h}(t)$:

$$h(t) = A(t)h(t-1) + Bx(t)$$
$$\overline{h}(t) = A(t)\overline{h}(t-1) + B(x(t) + \delta_x(t)).$$

By subtracting two systems, we have quantization error $\Delta(t) = \overline{h}(t) - h(t)$ given by: $\Delta(t) = A(t)(\overline{h}(t-1) - h(t-1)) + B\delta_x(t) = A(t) \cdot \Delta(t-1) + B\delta_x(t)$. Let's consider the recurrence on quantization error, assuming that $\Delta(0) = 0$:

$$\begin{aligned}
&\text{t=1, } \Delta(1) = A(1)\Delta(0) + B\delta_x(1) = B\delta_x(1) \\
&\text{t=2, } \Delta(2) = A(2)\Delta(1) + B\delta_x(2) = A(2)B\delta_x(1) + B\delta_x(2) \\
&\text{t=3, } \Delta(3) = A(3)\Delta(2) + B\delta_x(3) = A(3)A(2)B\delta_x(1) + A(3)B\delta_x(2) + B\delta_x(3) \\
&\text{t=4, } \Delta(4) = A(4)\Delta(3) + B\delta_x(4) = A(4)A(3)A(2)B\delta_x(1) + A(4)A(3)B\delta_x(2) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad + A(4)B\delta_x(3) + B\delta_x(4) \\
&\quad \ldots
\end{aligned}$$

At time step t, we have

$$\begin{aligned}
\Delta(t) &= A(t)\Delta(t-1) + B\delta_x(t) \\
&= \Pi_{u=2}^{t} A(u)B\delta_x(1) + \Pi_{u=3}^{t} A(u)B\delta_x(2) + \Pi_{u=4}^{t} A(u)B\delta_x(3) \\
&\quad + \ldots + A(t)B\delta_x(t-1) + B\delta_x(t) \\
&= \Sigma_{v=2}^{t}\big(\Pi_{u=v}^{t} A(u)B\delta_x(v-1)\big) + B\delta_x(t).
\end{aligned} \tag{4}$$

Let's consider quantization error $\Delta(t)$ such that $||\Delta(t)||_2 = ||\overline{h}(t) - h(t)||_2$:

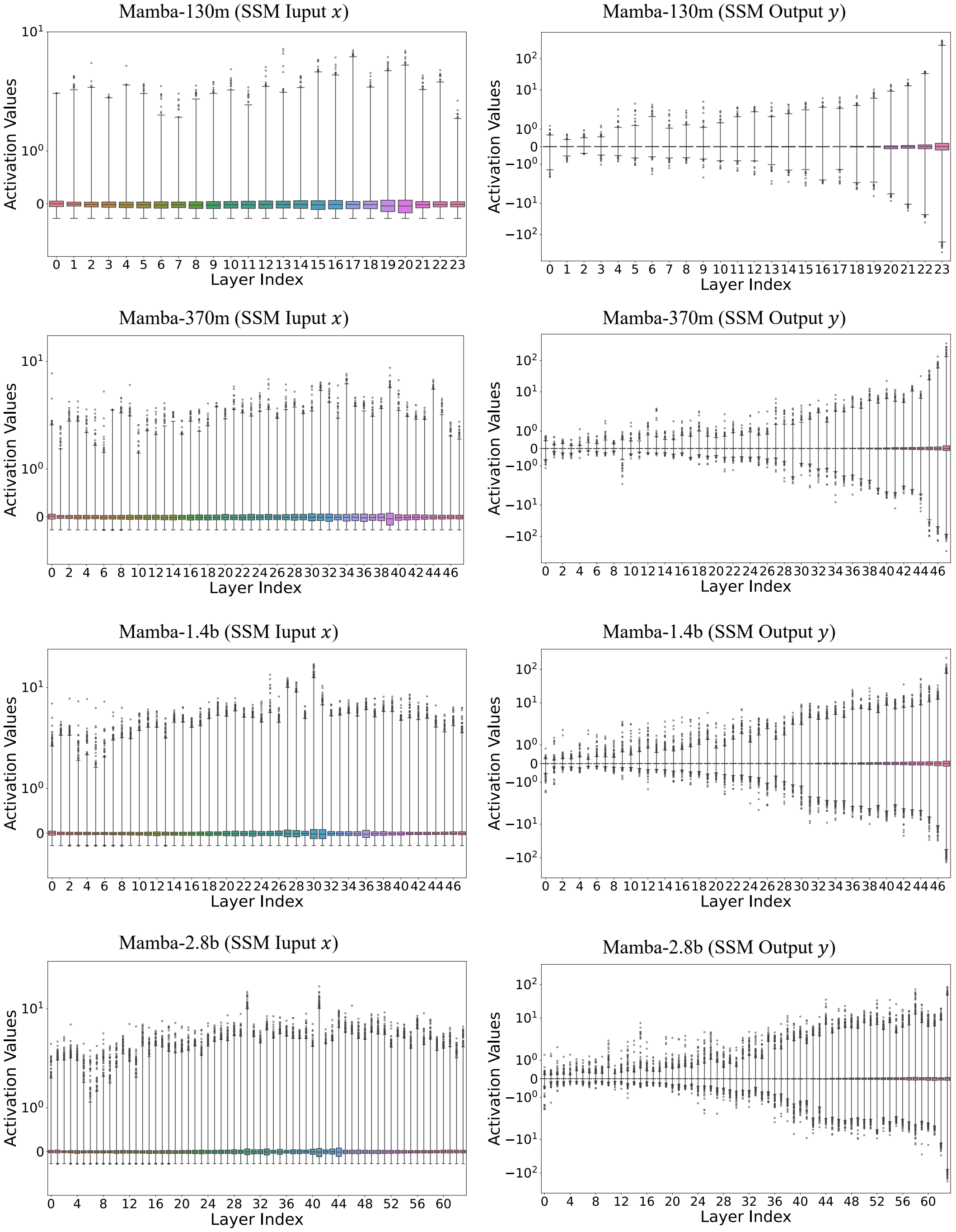
$$\begin{aligned}
||\Delta(t)||_2 &= ||\Sigma_{v=2}^{t}\big(\Pi_{u=v}^{t} A(u)B\delta_x(v-1)\big) + B\delta_x(t)|| \\
&\le ||\Sigma_{v=2}^{t}\big(\Pi_{u=v}^{t} A(u)B\delta_x(v-1)\big)|| + ||B\delta_x(t)|| \\
&\le ||\Sigma_{v=2}^{t}\big(\Pi_{u=v}^{t} A(u)B\delta_x(v-1)\big)|| + \epsilon b \\
&\le \epsilon b||\Sigma_{v=2}^{t}\big(\Pi_{u=v}^{t} A(u)\big)|| + \epsilon b \\
&\le \epsilon b \Sigma_{v=2}^{t}||\Pi_{u=v}^{t} A(u)|| + \epsilon b \\
&\le \epsilon b \Sigma_{v=2}^{t}\Pi_{u=v}^{t}||A(u)|| + \epsilon b \\
&\le \epsilon b \Sigma_{v=2}^{t}\Pi_{u=v}^{t} a e^{u-T} + \epsilon b \\
&\le \epsilon b (\Sigma_{v=2}^{t} a^{t-v+1}\Pi_{u=v}^{t} e^{v-T} + 1) \\
&= \epsilon b (\Sigma_{i=1}^{t-1} a^{i}\Pi_{j=0}^{i-1} e^{t-j-T} + 1) \\
&= \epsilon b (\Sigma_{i=1}^{t-1} a^{i} e^{(t-T)\cdot i}\Pi_{j=0}^{i-1} e^{-j} + 1) \\
&\le \epsilon b (\Sigma_{i=1}^{t-1}\left(\frac{ae^{t}}{e^{T}}\right)^{i} + 1) \\
&= \epsilon b (\frac{1 - (\frac{ae^{t}}{e^{T}})^{t}}{1 - (\frac{ae^{t}}{e^{T}})}) \\
&\le \epsilon b (\frac{1}{1 - ae^{t-T}})
\end{aligned}$$

The time-step dependent bound becomes: $||\Delta(t)||_2 \le \epsilon b(\frac{1}{1-ae^{t-T}})$. Therefore, the global bound for a given sequence length T (*i.e.*, $t = T$) is $||\Delta(T)||_2 \le \frac{\epsilon b}{1-a}$. $\square$

I LAYER-WISED DISTRIBUTION OF SSMS

We analyzed the input and output distributions of SSM for Mamba family, as shown in Figure 11. The box plots reveal the presence of outliers in both inputs and outputs.

SSM input x. In Figure 11 left, a skewed as well as asymmetric distribution is present in all layers of SSM inputs. Although the activation values are relatively small (< 10), however, due to the sensitivity of the linear recurrent dynamics, representing the input activations with low bit-width data causes significant performance drops for all sizes of Mamba-family. Therefore, clipping the distributions results in better quantization precision and avoids the accuracy degradation. Though asymmetric quantization performs slightly better than symmetric quantization as shown in Table 9, we choose symmetric quantization due to the support of most frameworks, *e.g.*, `CUTLASS`. We leave the asymmetric optimization in the future work.



Mamba-130m (SSM Iuput x)

Mamba-130m (SSM Output y)

Mamba-370m (SSM Iuput x)

Mamba-370m (SSM Output y)

Mamba-1.4b (SSM Iuput x)

Mamba-1.4b (SSM Output y)

Mamba-2.8b (SSM Iuput x)

Mamba-2.8b (SSM Output y)

Figure 11: Box plots of the SSM inputs and outputs distributions for Mamba family. We obtained the distributions using the Pile dataset, which serves as the calibration dataset for all our experiments.

SSM output y. In Figure 11 right, large outliers are observed in the SSM outputs. The outliers pose a great challenge in quantizing SSMs, due to the skewed quantization resolution caused by the extreme values ($\geq$ 100). This finding in SSMs echoes the outlier phenomenon observed in Transformers. We apply the Hadamard matrices to transform the output activations to an outlier-free space, making quantization easier. Interestingly, the layers closer to the model output have larger outlier values, suggesting that different quantization schemes can be applied to the earlier layers. We leave the study for future work.

J EXPLORE QUANTIZATION ALGORITHMS FOR SSM INPUTS

In our investigation of 8-bit quantization strategies for the state space model (SSM), we observed that the SSM input x exhibits outliers. Although these outliers are not excessively large, their presence significantly impacts the quantization error, affecting the SSM output y. In this section, we present the results and analysis of several other 8-bit quantization options in Table 9, though *not* necessarily hardware efficient.

Table 9: We explore different methods for quantizating SSM inputs into 8-bit. For the quantization of other tensors, we use the same settings as Quamba. We evaluate the quantized models and report the accuracy on the LAMBADA dataset. Our method is generalized to current mainstream freamworks and toolchains. (Sym., Asym., and Per. are short for symmetric, asymmetric, and percentile)

Methods for SSM Inputs	Framework Supports	LAMBADA Accuracy (↑)			
		130m	370m	1.4b	2.8b
FP16	Yes	41.24%	51.81%	63.82%	67.51%
Dynamic					
MinMax Sym.	Yes[3]	40.38%	51.45%	62.55%	66.62%
Static					
MinMax Sym.	**Yes**	34.10%	45.78%	44.89%	55.71%
MinMax Sym. Log2	No	40.31%	**51.80%**	**63.57%**	**67.51%**
MinMax Asym. Per.	No	**40.73%**	50.46%	63.09%	66.76%
MinMax Sym. Per. (Ours)	**Yes**	40.61%	50.37%	60.43%	65.67%

Dynamic quantization. One direct approach to provide a more accurate quantization mapping is through dynamic quantization. By dynamically capturing the activation range based on the current inputs, we can map the floating value into 8-bit with precise scaling factors. The approach boosts the accuracy and closes the performance gap between FP16. However, the dynamic approach will result in extra execution overhead on re-calculating the quantization scales, leading to sub-optimal computation efficiency.

Asymmetric quantization. We notice that the visualized tensor distribution of SSM inputs is asymmetric in Figure 11. To better utilize the bit-width, we could apply asymmetric quantization to the SSM inputs. As shown in Table 9, asymmetric quantization yields better accuracy in the zero-shot tasks, particularly for Mamba-1.4b and Mamba-2.8b models. However, asymmetric quantization increases the computational load during inference and requires specific software framework and hardware supports. We leave the asymmetric optimization for future work.

Log2 quantization. To avoid the quantization step skewed by the outliers, and to ensure that smaller values in a tensor are accurately mapped, we can quantify the tensor in log-scale. Here, we implement log2 quantization (Alsuhli et al., 2023; Miyashita et al., 2016), which maps the values to the nearest power of 2, achieving the desired non-uniform mapping. Our log2 version slightly outperforms Quamba. However, log2 matrix multiplication requires specific optimization on both software and hardware levels. Our method is more generalizable to mainstream frameworks and toolchains (*i.e.,* `PyTorch`, `CUTLASS`).

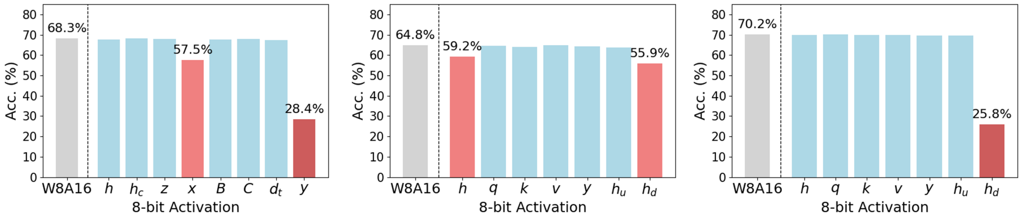
---

[3] The dynamic approach will result in extra execution overhead on re-calculating the quantization scales.

Other alternatives. Exploring the power of exponents with advanced low-bit floating-point data types (Kuzmin et al., 2024), such as E5M2 or E4M3 (Choquette, 2023), currently supported by NVIDIA Hopper GPUs, might also be an effective solution for quantizing SSM inputs. A more comprehensive study on quantizing SSMs using floating-point quantization is left for future work.

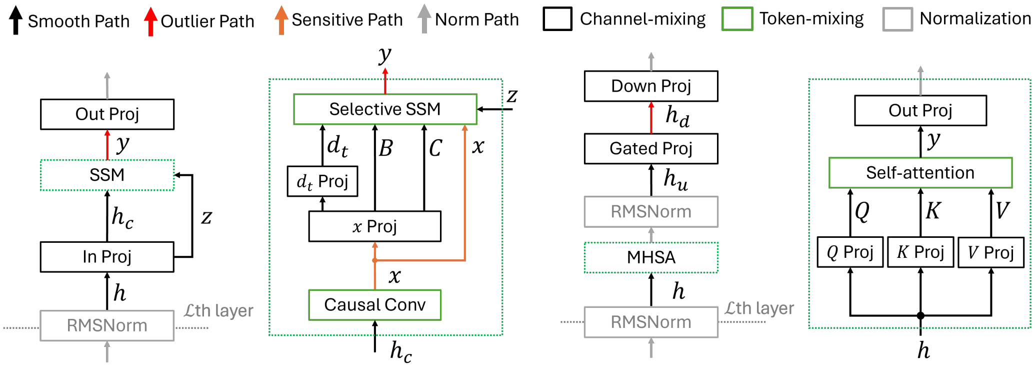
K COMPARING SSMS WITH SELF-ATTENTION LAYERS

Comparing the quantization sensitivity. We explore the quantization sensitivity of input and output activation maps for both SSMs and self-attention layers, as shown in Figure 12. We conduct experiments on Mamba 2.8B, a same-sized Transformer Pythia 2.8B, and a recently published, comparably sized Transformer Llama 3.2 3B. Quantizing the input x and output y tensors leads to the most significant accuracy drop on the LAMBADA dataset for SSMs, compared to other tensors. In contrast, quantizing the tensors in self-attention layers (*i.e.* h, q, k, v, and output y) results in minimal accuracy loss. For transformers, using 8-bit h_d tensors in feedforward layers significantly degrades model performance. Due to the differing quantization sensitivity patterns between Mamba and Transformer blocks, we highlight the *smooth*, *outlier*, and *sensitive* paths for both in Figure 13. This highlights that different quantization methods are needed for SSM-based models.



(a) Mamba 2.8B (b) Pythia 2.8B (c) Llama3.2 3B

Figure 12: Quantizing the input x and output y tensors causes the most significant accuracy drop on the LAMBADA dataset for SSMs. In contrast, quantizing the tensors in self-attention layers results in minimal accuracy loss.


Smooth Path
Outlier Path
Sensitive Path
Norm Path
Channel-mixing
Token-mixing
Normalization
Out Proj
y
SSM
h_c
z
In Proj
h
RMSNorm
ℒth layer
y
Selective SSM
z
d_t
B
C
x
d_t Proj
x Proj
x
Causal Conv
h_c
Down Proj
h_d
Gated Proj
h_u
RMSNorm
MHSA
h
RMSNorm
ℒth layer
Out Proj
y
Self-attention
Q
K
V
Q Proj
K Proj
V Proj
h

(a) The Mamba block (b) SSM (c) The Transformer block (d) MHSA

Figure 13: We annotate the different sensitivity patterns for SSMs and self-attention layers, which shows that they require different quantization techniques. The residual connections are simplified to avoid cluttering the figure.

Visualizing activations. We visualize these activation maps in Figures 14 and 15. Figure 14 shows the SSM activation maps in the last layer of Mamba 2.8B, while Figure 15 shows those for the self-attention layer in Llama 3.2 3B. We use the same test sample to visualize these activations.

Notably, SSMs and self-attention layers exhibit *distinct* activation patterns. Outliers appear in the SSM output y, unlike in self-attention layers, where the outputs y remain smooth and *do not* present

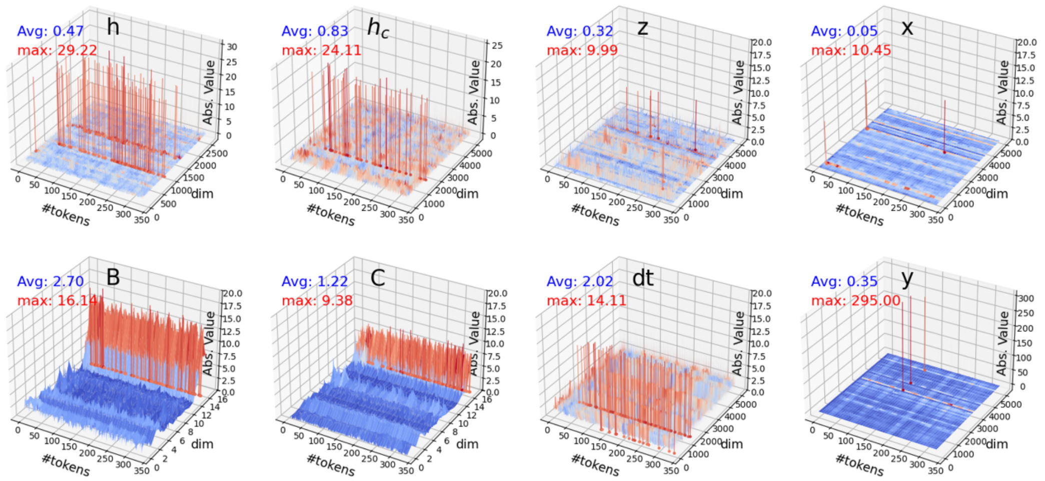


Figure 14: We visualize the activations of the SSM in the last layer of Mamba 2.8B. This figure shows that the values in the x tensor are not significantly large. However, outliers are present in the y tensor of the SSM output, making them difficult to represent in low bit-width data types.

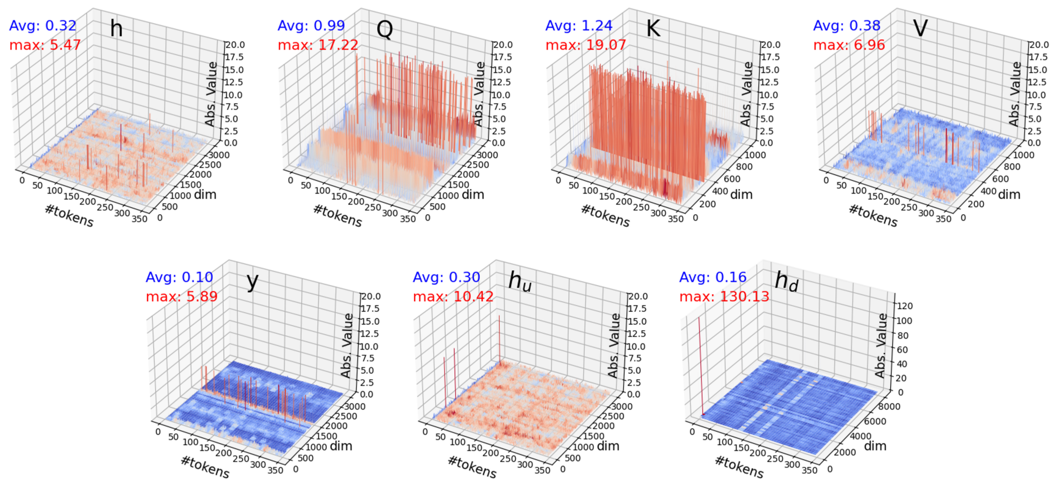
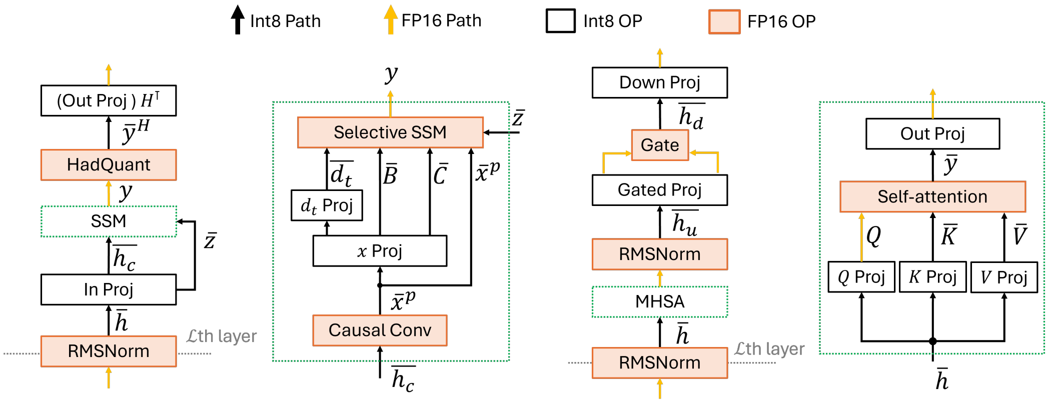


Figure 15: We visualize the activations of the self-attention layer in the last layer of Llama 3.2 3B. This figure shows that outliers are *not* present in the output y of the self-attention layers. In contrast to SSMs, the outlier issue arises in the h_d tensor of the *feedforward layers*.

outliers. In Transformer blocks, outliers only occur in the h_d of the feedforward layers, making them difficult to quantize. In contrast, Mamba blocks have large outliers in the SSM output y tensor, which are challenging to represent with low-bit data types. While the x tensor values are not significantly large, they are highly sensitive to quantization errors. We address this issue in Mamba blocks and introduce Quamba to manage the unique quantization patterns of SSMs.

Comparing the precision mapping. Figure 16 compares our precision mapping to the standard one used in Transformers (Zhao et al., 2023; Ashkboos et al., 2024b; Lin et al., 2024; Xiao et al., 2023).


Int8 Path
FP16 Path
Int8 OP
FP16 OP
(Out Proj) H^T
$\bar{y}^H$
HadQuant
y
SSM
$\bar{z}$
$\bar{h}_c$
In Proj
$\bar{h}$
RMSNorm
$\mathcal{L}$th layer
y
Selective SSM
$\bar{d}_t$
$\bar{B}$
$\bar{C}$
$\bar{x}^p$
d_t Proj
x Proj
Causal Conv
$\bar{h}_c$
Down Proj
$\bar{h}_d$
Gate
Gated Proj
$\bar{h}_u$
RMSNorm
MHSA
$\bar{h}$
RMSNorm
$\mathcal{L}$th layer
Out Proj
$\bar{y}$
Self-attention
Q
$\bar{K}$
$\bar{V}$
Q Proj
K Proj
V Proj
$\bar{h}$

(a) Precision Mapping of a Quamba block (Ours) (b) Precision Mapping of an 8-bit Transformer block

Figure 16: We compare our precision mapping to the precision mapping commonly used in Transformers. The residual connections are simplified to avoid cluttering the figure.

L LOW BIT-WIDTH QUANTIZATION

We show that Transformer-based quantization methods do not generalize well to Mamba blocks. We re-implement state-of-the-art low bit-width quantization methods for Transformers, Quip# (weight-only quantization, W2A16) (Tseng et al., 2024) and QuaRot (W4A4) (Ashkboos et al., 2024b), for the Mamba structure. These are denoted by Mamba-Quip# and Mamba-Had, respectively Our experiments show that they fail to effectively quantize Mamba in low bit-width settings, as shown in Tables 10 and 11. In Table 10, applying Quip# and QuaRot to Mamba results in much higher perplexity, leading to worse performance compared to Transformers. For instance, Mamba-Quip# quantizes Llama-2-7b with W2A16, causing only a 1.02× increase in perplexity, whereas our implementation on Mamba results in a 1.34× increase. In Table 11, although models with different bit-widths cannot be directly compared, the results show that both Mamba-Quip# and Mamba-Had reduce the average accuracy across six zero-shot downstream tasks. Our study highlights that quantizing SSMs is particularly challenging, as their input and output activations exhibit a causal relationship with varying levels of outliers, a phenomenon unique to SSMs (*ref.* Section K).

Table 10: Quantizing Llama-2-7b and Mamba 2.8B with low bit-width methods. The perplexity on Wiki2 is reported.

Methods	Precision	Llama-2-7b	Mamba-2.8b
Mamba	FP16	5.47	9.45
Mamba-Quip# Mamba-Had	W2A16 W4A4	5.56 (1.02 ×) 6.10 (1.11 ×)	12.71 (1.34 ×) failed
Quamba (Ours)	W8A8	-	**9.91**

Table 11: Quantizing Mamba 2.8B with low bit-width methods. The average accuracy on six zero-shot tasks is reported.

Methods	Precision	Mamba-2.8b
Mamba	FP16	63.1%
Mamba-Quip# Mamba-Had	W2A16 W4A4	58.5% 30.2%
Quamba (Ours)	W8A8	**62.2%**

M LIMITATIONS AND BROADER IMPACTS

We find that the accuracy degradation is not negligible in both accuracy and perplexity in Table 7 and Table 2. Despite this, the performance trade-off is acceptable given the significant improvements in latency and resource efficiency. Our work enables large language models to be deployed on resource-limited devices. As a positive feature, our method may push the development of privacy-centric on-device applications, where sensitive data can be processed locally without relying on cloud services. However, our work may also present challenges such as increased device resource consumption and potential security vulnerabilities if the local devices are compromised.