SSM Adapters via Hankel Reduced-order Modeling: Injection Site Determines Task Suitability in Long-Context Fine-Tuning

Anonymous Authors[1]

Abstract

While parameter-efficient fine-tuning (PEFT) typically targets attention projectors, its efficacy for tasks requiring sequential state accumulation remains under-explored. We examine if PEFT for such tasks can benefit from state space model (SSMs) adapters, and if MLP blocks are better injection sites. We introduce Hankel Reduced order Model (HRM) adapter, an SSM-based residual module initialized via Balanced Truncation of empirical Hankel Grammians. By leveraging the time-invariance of the system matrix $\bar{A}$, HRM enables an exact FFT-based parallel scan, achieving computational parity with LoRA across all context lengths. In iso-parametric evaluations on Mistral-7B (8.4M trainable parameters), HRM outperforms LoRA variants on LongBench tasks, including QuALITY (+34.8% relative accuracy) and QMSum (+71.6% relative ROUGE-1). HRM further demonstrates consistent superiority across 18 configurations of synthetic state-tracking (DFA, Parity) and character-level language modeling (enwik8). Gate analysis reveals that HRM adapters effectively learn to modulate recurrence, providing a robust architectural alternative to low-rank adaptation for long-context sequence modeling.

1. Introduction

Parameter-efficient fine-tuning (PEFT) is a dominant paradigm in adapting large pre-trained language models (LLMs) for downstream tasks. Rather than updating entire model weights, PEFT methods insert adapters or modify a smaller subset of parameters, keeping the model backbone frozen. Low-Rank Adaptation (Hu et al., 2022) is the most widely adopted PEFT method, achieving strong results across language understanding, generation, and instruction following tasks while adding ∼0.1–1% extra parameters. LoRA parameterizes a weight update as $\Delta W = BA$, where $B \in \mathbb{R}^{d_{out} \times r}$ and $A \in \mathbb{R}^{r \times d_{in}}$. The matrices A, B are learned such that rank $r \ll \min(d_{out}, d_{in})$, where the full forward pass through the adapted layer becomes:

$$h_t = W_0 x_t + BAx_t = (W_0 + BA)x_t \tag{1}$$

where the model weights W_0 are kept frozen for input x_t, at position t. We observe that the computation to adapt weights in LoRA (and its related methods: DoRA (Liu et al., 2024), QloRA (Dettmers et al., 2023), AdaLoRA (Zhang et al., 2023b)) is a static linear function of the input x_t. As a result, the adapter output at position t has no access to the prior positions: $x_{t-1}, x_{t-2}, \cdots$ .. This is not a failure of reduced rank modeling as no choice of r will give LoRA temporal memory access.

To motivate this central issue, consider fine-tuning a model to simulate a 4-state Deterministic Finite Automaton (DFA). At each step, the correct output depends not on the current input symbol alone, but on the accumulated sequence of transitions since the start. A DFA with 4 states can be in any of 4 configurations depending on the entire history $x_1, x_2, \cdots, x_{t-1}$. LoRA, regardless of rank, collapses the current state to a static function of x_t, therefore, it structurally cannot represent a state that persists across positions. Despite this, LoRA, DoRA, AdaLoRA, and QLoRA's successes in achieving excellent results on tasks where adaptation is position-independent, such as domain style transfer, factual knowledge injection, and instruction following is well established in literature.

To this end, we investigate the following question: *is it possible to construct a PEFT adapter that (1) adds temporal recurrent state to a frozen transformer, (2) is provably compressible to a minimal state dimension, (3) and is computationally equivalent to LoRA, while achieving better performance on long-range tasks across diverse domain?*

2. Related Works

All major **PEFT** methods share the common structural property of position-independence (or position agnostic weight fine-tuning). LoRA (Hu et al., 2022), AdaLoRA (Zhang

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

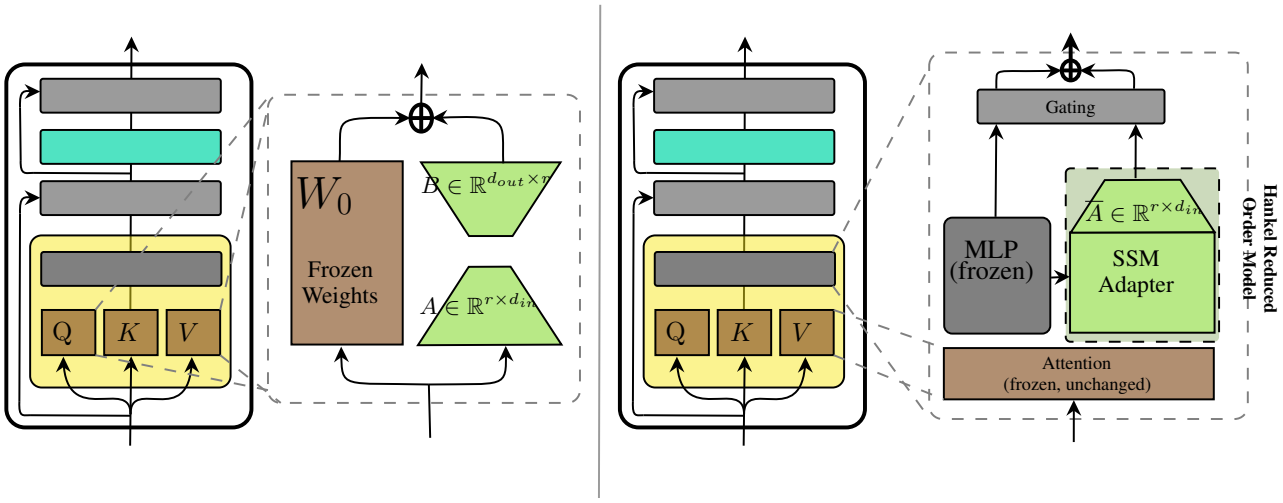

Figure 1. Architecture comparison. LoRA modifies weight matrices; its output at position t is a static function of x_t. The HRM adapter inserts a parallel recurrent branch whose hidden state integrates all prior representations.

et al., 2023b), QLoRA (Dettmers et al., 2023), LoRA+ (Hayou et al., 2024): all compute $h = f(x_t)$ with no dependence on t or prior positions. AdaLoRA adaptively allocates rank but the resulting update is still a static matrix product. IA^3 (Liu et al., 2022) applies learned vectors to rescale hidden states, a multiplication by a position-independent scalar, also resulting a static update (see Fig. 1).

Some works prepend learned soft tokens to the input (Lester et al., 2021; Li & Liang, 2021). These tokens provide context at the input but do not define a recurrent state, and the transformer still processes each position independently after the prefix. Foundational adapter methods for PEFT (Houlsby et al., 2019; Pfeiffer et al., 2020) insert small MLP bottlenecks to pre-trained transformer models. The bottleneck $h = W_2 \cdot \sigma(W_1 \cdot x_t)$ depends only on x_tm without memory recurrence. While (Houlsby et al., 2019) places two adapter bottleneck modules into each transformer layer, while (Pfeiffer et al., 2020) places a single adapter, halving the number of trainable parameters.

On the other hand, State Space models (SSMs) have been shown promise to alleviate the quadratic attention costs over long-contexts. Structured State Space Sequence (S4) Models were introduced by (Gu et al., 2021) with state space layer with HiPPO-based initialization and convolution-mode inference, which was improved in S4D (Gu et al., 2022) by restricting to diagonal state space matrix $\bar{A}$, losing expressiveness but enabling simpler inference. Finally, Mamba models (Gu & Dao, 2023) introduced input-dependent state matrices (A_t, B_t), enabling selective memory.

Hybrid model architectures such as Griffin (De et al., 2024), MambaFormer (Park et al., 2024), and Jamba (Lieber et al., 2024) utilize SSM layers with transformers. On the surface they seem similar to our proposed work (HRM inserts a $d = 32$ SSM at each MLP block; MambaFormer inserts Mamba layers between attention blocks). The critical distinction is one of training regime: every hybrid architecture requires joint training from scratch on billions of tokens. HRM is the first method that adds SSM-style temporal memory in the PEFT setting; therefore, the backbone is frozen, the adapter has $\sim 0.1\%$ parameters, and no pre-training data beyond the fine-tuning task is required. As a result, a user with a frozen, pre-trained GPT-2 cannot apply MambaFormer to it, but they can apply HRM.

The combination of (a) recurrent hidden state, (b) provable compression via model-order reduction, and (c) computational parity with static adapters does not appear in the literature. To the best of our knowledge, the closest related work is SLoRA ((Sheng et al., 2024)) and related low-rank SSM approaches that treat SSMs as a structured alternative to LoRA rank approximations. However, these do not apply Balanced Truncation, do not provide error bounds, and do not address the computational overhead of the recurrence.

3. Background

LoRA Low-Rank Adaptation (LoRA) (Hu et al., 2022) relies on the observation that weight updates during fine-tuning $\Delta W \in \mathbb{R}^{d_{out} \times d_{in}}$ of pre-trained models lie in a low intrinsic dimension (Aghajanyan et al., 2021). This motivates parameterizing the update as a rank-r product:

$$\Delta W = BA, B \in \mathbb{R}^{d_{out} \times r} \text{ and } A \in \mathbb{R}^{r \times d_{in}} \quad (2)$$

During training W_0 is frozen, and only B and A are updated. At inference, the update is absorbed as $W_{eff} = W_0 + BA$, adding no inference latency, with the forward pass:

$$h = W_{eff}x = (W_0 + BA)x = W_0x + B(Ax) \quad (3)$$

LoRA is applied to the Q and V projection matrices of each self-attention block in standard practice. For a model with n_{layers}, d_{model} attention dimension, this contributes $4r \cdot n_{layers} \cdot d_{model}$ trainable parameters. The mapping $h = (W_0 + BA)x$ is a linear function of x alone. The matrix W_{eff} is fixed at all positions. If we index the sequence position as t, the LoRA output at position t is $h_t^{LoRA} = (W_0 + BA)x_t$ with no dependence on the previous inputs x_{t-1}, etc.

The adapter applies the same linear transformation B A to every token, independently of position or context, and is therefore memory less. AdaLoRA (Zhang et al., 2023b) addresses rank allocation but not memory either. It parameterizes $\Delta W = P\Lambda Q$ where P, Q are orthogonal and Λ is diagonal (singular value decomposition structure), pruning entries of Λ based on importance. The result is still a static linear map of the current token. QLoRA (Dettmers et al., 2023) addresses memory efficiency (4-bit quantization of $W0$) and DoRA (Liu et al., 2024) decomposes into magnitude and direction components. Both remain static functions of the current token. The memory-less property is therefore preserved in existing LoRA variants.

SSMs A continuous-time linear state-space model (SSM) is defined by the equations:

$$\dot{x}(t) = Ax(t) + Bu(t),\ y(t) = Cx(t) + Du(t) \tag{4}$$

for hidden state $x \in \mathbb{R}^d$, input $u \in \mathbb{R}^m$, output $y \in \mathbb{R}^p$, and the state-transition (or system) matrix A, B the input matrix, C the output matrix, and D the feed-through, or skip matrix. For sequence modeling, the continuous-time system is discretized to obtain a recurrence relation. Given a time step Δt, the Zero-Order Hold (ZOH) discretization yields:

$$\begin{aligned} x_t &= \bar{A}x_{t-1} + \bar{B}u_t,\ y_t = Cx_t \\ \bar{A} &= e^{A\Delta t}, \bar{B} = A^{-1}(e^{A\Delta t} - I)B \end{aligned} \tag{5}$$

The discrete SSM defines a linear map from the input sequence $\{u_1, ..., u_T\}$ to the output sequence $\{y_1, ..., y_T\}$:

$$y_t = \sum_{k=0}^{t} \underbrace{C\bar{A}^{t-k}\bar{B}}_{g_{t-k}} u_k = (g \star u)_t \tag{6}$$

where g_k is the impulse response of the system. This results in the output sequence be written as the causal convolution of the impulse response with the input.

Structured State Spaces (S4) (Gu et al., 2022) showed that when $\bar{A}$ is initialized as a specific Normal Plus Low-Rank (NPLR) matrix, the SSM can model long-range dependencies with a stable impulse response that decays slowly. The key computational insight of S4 is that the causal convolution $(g \star u)$ can be computed in $\mathcal{O}((T \log T))$ via FFT. We will also use this fact for our computation in the subsequent sections.

Finally, the *stability of the discrete SSM* in (5) requires all eigenvalues of $\bar{A}$ to lie strictly within the unit circle, i.e., $\max \left|\lambda_i(\bar{A})\right| < 1$. For diagonal $\bar{A}$ with real entries, this requires $\left|\bar{A}_{ii}\right| < 1$. We will enforce this by parameterization, as our reduced-order modeling requires stability of the underlying linear time invariant (LTI) system.

Balanced Truncation in LTI Systems Consider LTI dynamics (i.e., fixed $G \triangleq (\bar{A}, \bar{B}, C, D)$) in (5), with state dimension d. The reduced order modeling problem for LTI dynamical system G is then to find a reduced order system $\hat{G}$ with state dimensions $\hat{d} < d$, such that the input-output behaviors of G and $\hat{G}$ are as close as possible, with a quantified error bound. *Balanced Truncation* (BT) (Moore, 2003) is the canonical solution to this problem for stable LTI systems.

The LTI system's state $v \in \mathbb{R}^d$ is *controllable* if there exists an input sequence to drive G from the origin to v. Controllability for the LTI system relies on the *Controllability Grammian* $W_c \in \mathbb{R}^{d\times d}$, a positive semi-definite matrix defined as:

$$W_c = \sum_{k=0}^{\infty} \bar{A}^k \bar{B}\bar{B}^T (\bar{A})^k \tag{7}$$

Equivalently, W_c is known to be the solution of the discrete time Lyapunov equation (Corless & Frazho, 2003):

$$\bar{A}W_c\bar{A}^T - W_c + \bar{B}\bar{B}^T = 0 \tag{8}$$

Conversely, the state v is *observable* if the initial state v can be uniquely determined from the output sequence $\{y_k\}$. Similarly, observability for the LTI system relies on its *Observability Grammian* $W_o \in \mathbb{R}^{d\times d}$, defined as:

$$W_o = \sum_{k=0}^{\infty} (\bar{A}^T)^k C^T C \bar{A}^k \tag{9}$$

with its corresponding Lyapunov equation:

$$\bar{A}^T W_o \bar{A} - W_o + C^T C = 0 \tag{10}$$

Matrices W_c and W_o play an important role in balanced truncation of G by forming a joint Hankel operator $\mathcal{H}$: {past inputs} $\rightarrow$ {future outputs}. For discrete time LTI system, $\mathcal{H}$ is the fixed matrix $\Gamma \triangleq W_c W_o$, which maps the full causal history of inputs to all future outputs. To perform truncation, we need to align the coordinate system so that directions are ordered by their joint controllability/observability. The diagonal entries σ_i's are called the *Hankel singular values* (HSVs):

$$\sigma_i = \sqrt{\lambda_i(W_c W_o)},\ \ \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d \geq 0 \tag{11}$$

also the ith singular value of the Hankel operator, characterizing a state direction that is irrelevant to the past-to-future input-output map. The balancing transformation is a coordinate transform $T \in \mathbb{R}^{d\times d}$ such that the Grammians are simultaneously diagonalized:

$$TW_cT^T = T^{-T}W_oT^{-1} = \Sigma = \text{diag}(\sigma_1, \cdots, \sigma_d) \quad (12)$$

As a result, the transformed system $(T\bar{A}T^{-1}, T\bar{B}, CT^{-1})$ has the property that each state direction has equal controllability and observability, equal to σ_i. Such a system is called a balanced system.

A balanced truncation $\hat{G}$ of the system G can now be formed by partitioning the balanced system into "important" $(1, \cdots, \hat{d})$ and "unimportant" blocks $(\hat{d}+1, \cdots, d)$ as:

$$\begin{aligned} \hat{A} &= \left(T_{[1:\hat{d},1:\hat{d}]}\bar{A}T^{-1}_{[1:\hat{d},:]}\right)_{[1:\hat{d},1:\hat{d}]}, \\ \hat{B} &= \left(T\bar{B}\right)_{[1:\hat{d},:]}, \hat{C} = \left(CT^{-1}\right)_{[:,1:\hat{d}]} \end{aligned} \quad (13)$$

Finally, Glover's error bound (Glover, 1984) dictates that the truncated system deviates from the original by at most twice the sum of the discarded HSVs:

$$\left\|G - \hat{G}\right\|_{\mathcal{H}_\infty} \leq 2 \sum_{k=\hat{d}+1}^{d} \sigma_k \quad (14)$$

This is a worst case bound over all inputs and all frequencies. Furthermore, $\hat{G}$ is stable, and Glover's bound is tight.

4. Method: Hankel-Reduced order Model Adapter

Empirical Grammians for SSMs HSV-based balanced truncation requires that system G be LTI. The proposed HRM adapter's $\bar{A}$ is time-invariant, so the theorem applies directly. However, the selective SSM extension (input-dependent B_t, C_t in selective scan (Gu & Dao, 2023)) violates the LTI assumption. We address this extension via *empirical Grammians* approach (Lall et al., 1999). This extends balanced truncation to time-varying systems by approximation Grammians from observed state trajectories. This involves running the system forward on a representative calibration dataset of N sequences. At each time step t of each sequence n, we record the state vectors $s_{n,t}$, to compute the empirical controllability Grammian as:

$$W_c^{emp} = \frac{1}{N \cdot t_{cal}} \sum_{n=1}^{N} \sum_{t=1}^{t_{cal}} s_{n,t} s_{n,t}^T \in \mathbb{R}^{d\times d} \quad (15)$$

This estimates the covariance of state trajectories under typical inputs, a proxy for controllability. A similar proxy for observability is found in the form of empirical observability Grammian as:

$$W_o^{emp} = \frac{1}{N \cdot t_{cal}} \sum_{n=1}^{N} \sum_{t=1}^{t_{cal}} y_{n,t}^T y_{n,t} \in \mathbb{R}^{d\times d} \quad (16)$$

Due to (Lall et al., 1999), $W_c^{emp} \to W_c$ and $W_o^{emp} \to W_o$ as $N \to \infty$, and the converges at $\mathcal{O}\left(1/\sqrt{N}\right)$.

For our case, this gives a $\mathcal{O}\left(N \cdot T_{cal} \cdot d^2\right)$ to compute HSVs, from which the balancing transform and truncation proceed exactly as in the LTI case. However, for the time-invariant HRM adapter, both the analytical Lyapunov and empirical Grammian approaches are available.

HRM Adapter Architecture Now we are ready to architect the HRM adapter based on Hankel order-reduction for our SSM. Consider a standard pre-norm transformer layer l with input $x_t \in \mathbb{R}^{d_{model}}$. The layer applies self-attention followed by an MLP sublayer, each with residual connections and layer normalization:

$$\begin{aligned} a_t &= x_t + \text{Attm}(\text{LayerNorm}(x_t)), \\ h_t^{MLP} &= a_t + \text{MLP}(\text{LN}(a_t)) \end{aligned} \quad (17)$$

All weights Attn(•) and MLP(•) are frozen during adapter training. The HRM adapter is inserted parallel to the lth MLP sub-layer, adding a recurrent correction to the MLP output:

$$h_t^{out,(l)} = h_t^{MLP,(l)} + \alpha^{(l)} \cdot y_t^{(l)} \quad (18)$$

where $\alpha^{(l)} \in \mathbb{R}$ is a layer-specific learnable gate scalar and $y_t^{(l)}$ is the adapter output for layer l, at position t.

The adapter at layer l defines a recurrent hidden state $s_t^{(l)} \in \mathbb{R}^d$ that integrates the token representations $h_1, h_2, \cdots, h_t$ as they are processed:

$$s_t^{(l)} = \bar{A}s_{t-1}^{(l)} + \bar{B}^{(l)}h_t^{MLP}, \;\; y_t^{(l)} = C^{(l)}s_t^{(l)} \in \mathbb{R}^{d_{domel}} \quad (19)$$

where $\bar{A} \in \mathbb{R}^{d\times d}$ is a learnable (diagonal) state transition matrix, $\bar{B} \in \mathbb{R}^{d\times d_{model}}$ maps the current hidden state into the adapter's state space, $C \in \mathbb{R}^{d_{model}\times d}$ maps the adapter state back to the hidden representation, and α is a learnable scalar gate. Using (18) and (19) the adapter's output can be unrolled as:

$$y_t^{(l)} = C^{(l)} \sum_{k=0}^{t} \left(\bar{A}^{(l)}\right)^{t-k} \bar{B}^{(l)} h_k^{MLP,(l)} \quad (20)$$

Therefore, the combined gated addition (18) gives the unrolled layer computation as:

$$h_t^{out,(l)} = h_t^{MLP,(l)} + \alpha^{(l)} C^{(l)} \sum_{k=0}^{t} \left(\bar{A}^{(l)}\right)^{t-k} \bar{B}^{(l)} h_k^{MLP,(l)} \quad (21)$$

The adapter is placed parallel to the MLP, with a learnable scalar weight α. There are two reasons to this. First, the attention mechanism already computes a weighted sum over all past positions, providing global context. Adding a recurrent branch to attention would interact with the causal mask in a non-trivial way and could disturb the attention distribution. Second, the MLP sub-layer is the natural site of position-independent computation as it applies the same learned function to each token representation independently. A recurrent adapter at this site adds the missing dependence on prior positions. On the other hand, a sequential insertion would mean the MLP receives adapter-modified input, potentially causing large gradient flows through the frozen MLP. The parallel insertion ensures the frozen MLP is always evaluated on the original attention output, while the adapter's contribution is additive and controlled by α. The gating scalar α is initialized to a small value to ensure that at the start of the training the HRM adapter contribution is small, and the model starts from the behavior of a pre-trained backbone. The gate then proceeds to grow with training and the adapter learns a useful temporal correction. Experiments show that initializing at $\alpha_0 = 1.0$ causes divergence on all tested configurations.

Another design choice we made was to have time-invariant $\bar{A}$. A natural concern is that Mamba-style selective SSMs (with input-dependent $\bar{A}_t, B_t, C_t$) are more expressive as they can selectively forget irrelevant tokens by adjusting the state decay on the fly. We fix in our architecture a time-invariant $\bar{A}$ deliberately because the pre-trained model's attention already handles selectivity to a certain degree. The frozen self-attention mechanism performs global, content-based retrieval at every layer, choosing which past tokens to attend to. The HRM adapter's role is complementary: it provides a continuous recurrent state that integrates the local MLP output stream, accumulating context that attention's position-independent MLP stream cannot represent. Further, time-invariant $\bar{A}$ makes $\bar{A}^k$ a geometric sequence, enabling the exact FFT convolution shortcut This eliminates the compute overhead and makes the HRM adapter practical, and temporal causal. Time-invariance also makes the trained adapter an LTI system, for which Balanced Truncation with the Glover $\mathcal{H}_\infty$ bound applies analytically. This also allows for an easier computation of Grammians using Lyapunov equation. The expressivity trade-off is knowingly accepted in exchange for theoretical tractability and computational efficiency.

HRM State Transition Matrix Parameterization & Stability We parameterize the state transition matrix $\bar{A}$ in (19) as a fixed diagonal (this is common in Mamba and S4D like SSMs (Gu & Dao, 2023; Gu et al., 2022)) to keep matrix-vector product cost $\mathcal{O}(d)$, instead of $\mathcal{O}(d^2)$ for a general $\bar{A}$:

$$\bar{A}^{(l)} = \operatorname{diag}\left(\bar{a}_1^{(l))}, \cdots, \bar{a}_d^{(l))}\right), \bar{a}_i^{(l))} = \exp\left(-\exp\left(\log A_i^{(l)}\right)\right) \tag{22}$$

where $\log A_i^{(l)} \in \mathbb{R}$ is the raw (unconstrained) learnable parameter. This parameterization is to ensure that $0 < \bar{a}_i^{(l)} < 1$, therefore, $\max_i \left|\lambda_i(\bar{A}^{(l)})\right| < 1$ for each layer. As a result, the HRM dynamics are unconditionally stable for any parameters values during all stages of training.

The parametrization above has a ZOH interpretation, as $\bar{A} = \exp(A\Delta t)$ for a continuous time system in (4) with ZOH discretization factor Δt. The parameter $\bar{a}_i = \exp(-\exp(\log A_i))$ corresponds to $\Delta t \cdot |A_i| \in (0, \infty)$, with $\exp(\log A_i)$ playing the role of $\Delta t \cdot |A_i|$. The combined parameter $\log A_i$ absorbs both, the magnitude or the continuous A eigenvalue, and the discretization step Δt. $\bar{B}$ and C are unconstrained dense matrices with learnable parameters $\bar{B}^{(l)} \in \mathbb{R}^{d \times d_{model}}$, and $C^{(l)} \in \mathbb{R}^{d_{model} \times d}$, both initialized with small-variance Gaussian entries. Finally, a learnable parameter $\log \Delta t^{(l)}$ associated with each layer l is used to compute the ZOH discretization step $\Delta t^{(l)}$. This allows the adapter to learn an appropriate timescale for the task. As a result, the total number of learnable parameters per layer are: $B : d \times d_{model}$, $C : d_{model} \times d$, $\log A_i : d$, $\log \Delta t : d$, and gate $\alpha : 1 = 2d \cdot d_{model} + 2d + 1$ parameters. Therefore, total number of parameters (compared with LoRA parameters) are:

$$\begin{aligned} P_{HRM} &= n_{layers} \cdot (2d \cdot d_{model} + 2d + 1) \\ P_{LoRA} &= n_{layers} \cdots 2 \cdots 2 \cdot r \cdot d_{model} \end{aligned} \tag{23}$$

Following this, a state compression is done so that for a fixed r, so that the iso-parametric P_{HRM} is compressed for a Glover bound $\varepsilon = 0.01$. That is, top 90% of the HSVs are kept, and the remaining discarded. This HSV-based compression means that $P_{HRM}(\hat{d}) \leq P_{HRM} \approx P_{LoRA}$.

Parallel Scan for HRM Adapter Since our $\bar{A}$ is time-invariant, the HRM recurrence computes a causal linear convolution. This convolution can be evaluated in $\mathcal{O}(T \log T)$ via the Fast Fourier Transform, replacing $\mathcal{O}(T)$ sequential Python-level dispatches with three FFT calls and achieving empirical compute parity with LoRA at all tested context lengths. This parity is shown in the compute wall clock times for HRM and LoRA in Appendix. B.

Using FFT-based parallel scan arguments from (Gu & Dao, 2023; 2024), let $\{h_t\}_{t=0}^{T-1} \subset \mathbb{R}^{d_{model}}$ be the input sequence to the HRM adapter. Let s_t and y_t be the HRM sequential recurrence state and output, respectively, for $s_0 = 0$. For the impulse response g_k defined in (5), define the zero-padded

sequences:

$$\tilde{g}[k] = \begin{cases} g_k & 0 \le k \le T-1 \\ 0 & T \le k \le 2T-1 \end{cases}$$
$$\tilde{h}[k] = \begin{cases} h_k & 0 \le k \le T-1 \\ 0 & T \le k \le 2T-1 \end{cases} \tag{24}$$

Then the output sequence can be computed as:

$$y_t = \sum_{k=0}^{t} t_{t-k} h_k = \left[\mathrm{IFFT}\left(\mathrm{FFT}(\tilde{g} \odot \mathrm{FFT}(\tilde{h}))\right)\right]_t \tag{25}$$

for $0 \le t \le T-1$, for element-wise product $\odot$, and FFT/IFFT operating on the length $2T$ dimensional sequence. Since all operations in the FFT scan (`torch.fft.rfft`, element-wise multiply, `torch.fft.irfft`) are differentiable in PyTorch, gradients flow back through the FFT to $\bar{A}, \bar{B}, C$ without any custom CUDA kernels.

5. Experiments

5.1. Synthetic Task: DFA State Tracking

Consider a deterministic finite automaton (DFA), a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ an input alphabet, $\delta : Q \times \Sigma \to Q$ the transition function, q_0 the initial state, and $F \subseteq Q$ the set of accepting states. Suppose we are given an input sequence $(\sigma_1, \cdots, \sigma_T) \in \Sigma^T$, a DFA with k states and binary alphabet $\Sigma = \{0, 1\}$. The state tracking task is to predict the current DFA state at each position t, given $(\sigma_1, \cdots, \sigma_T)$, predict $q_t = \delta(q_{t-1}, \sigma_t)$.

This requires exact state accumulation, since q_t depends on the full history $(\sigma_1, \cdots, \sigma_t)$ through the transition functions. A model that cannot maintain state across positions will fail as T grows as it must somehow compress the DFA state into the current token representation alone.

We experiment with $k \in \{2, 4, 8\}$, binary alphabet, context lengths $T \in \{64, 128, 256, 512\}$, with each DFA instance having a random fixed transition table δ. The model must output a k-way classification at each position, with sequences sampled uniformly at random from all valid DFA paths, 10,000 training sequences (1,000 validation sequences), across 3 different seeds.

In Fig. 2, we observe that HRM-BT dominates LoRA at all T values, and the gap grows with T, consistent with the memory hypothesis. Additionally, balanced truncation outperforms no truncation due to BT regularization. The HSV decay curve σ_i/σ_1 drops below 0.01 by $i = 7$, justifying $\hat{d}$=6. The HSV spectrum confirms that *DFA dynamics are intrinsically 6-dimensional, despite training with d=32 state dimensions*.

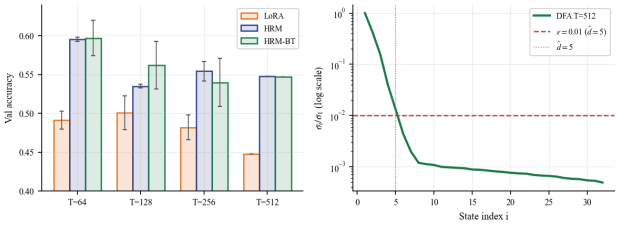

Figure 2. DFA state tracking results: (left) HRM vs. HRM with balanced truncation vs. LoRA, (right) Hankel Singular Value decay rate for the task, with HSV cutoff threshold = 0.01

6. MAESTRO Piano Language Modeling

MAESTRO v2 (Hawthorne et al., 2018) is a dataset of ∼200 hours of professional piano performances in symbolic MIDI format. We treat it as a character-level language modeling task: each MIDI event (note-on, note-off, time-shift, velocity) is encoded as a single token, and the model is trained to predict the next token given the context. The vocabulary has ∼300 distinct event tokens. Piano music is an ideal testbed for long-range temporal modeling: (1) melodic phrases span dozens to hundreds of notes; (2) harmonic progressions follow conventions (ii-V-I, etc.) that span 8–16 measures; (3) rhythmic structure repeats at multiple timescales. A model that can only attend to recent tokens (or a static adapter at each position) will fail to capture these structural regularities. Finally, audio processing tasks are generally suitable for SSM models over transformers, a benefit we expect to observe in the HRM adapter.

For this task, we had a backbone frozen TinyGPT (4-layer, d_model=128), context length T=512 events, with 80K events training, 5K validation.

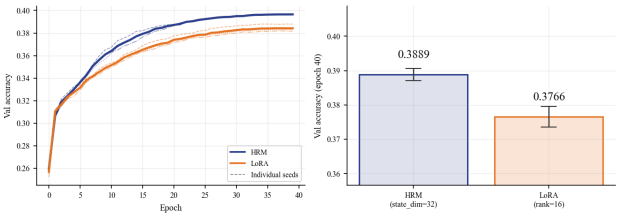

Figure 3. MAESTRO piano language modeling. (left) HRM vs. LoRA, (right) Final BPC.

The results are shown in Fig. 3. HRM achieves a lower BPC at convergence and with substantially smaller variance across seeds (band is nearly invisible for HRM). (Right) HRM accuracy 0.3966±0.0003 vs LoRA 0.3843±0.0033 at epoch 40, t=7.01, p¡0.001. Both adapters use identical parameter budgets (Tier 2, 33K parameters).

6.1. `enwik8` Character Language Modeling

`enwik8` is a widely used character level language modeling consisting of 100 million bytes of XML formatted Wikipedia text (Mahoney, 2013). The task is to minimize bits-per-character (BPC), the number of bits required to encode each character on average. Since English text has word-level, sentence-level, and paragraph-level structure, `enwik8` is appropriate to test long-range adapter capability for PEFT tasks. Standard benchmarks use context lengths of 512–8192 characters, which are long by transformer standards.

For comparing HRM with LoRA, we utilize a backbone frozen TinyGPT (4-layer, d_{model} =128), with context lengths $T \in \{512, 1024, 2048\}$, 3 tiers of model capacity (r and $\hat{d}$ values. Due to different convergence rates, we use 25 epochs for $T = 512$ and 40 epochs for $T = 1024$ and 2048, with a batch size of 32, 10,000 training examples, and 1,000 validation examples. The findings across these cases are in Table 3 with HRM achieving a lower BPC than LoRA adapter on *all configurations*. The BPC-T relation is detailed in Appendix D.

Remark 6.1. Our model (≈1.1M backbone params + 33K adapter) is not competing with full-scale `enwik8` models. Transformer-XL (44M params) achieves 1.06 BPC; our goal is not SoTA but the relative difference ΔBPC = BPC(LoRA)-BPC(HRM) as T varies. A positive growing ΔBPC demonstrates that the HRM can help improve even small capacity TinyGPT backbone on the task.

7. Mistral-7B LongBench

In this final experiment, we evaluate HRM against four different LoRA family baselines: LoRA, AdaLORA, DoRA, and QLoRA. LongBench is a comprehensive benchmark for evaluating LLMs on their ability to understand and process long-context information across various tasks (Bai et al., 2024). We do this over three different LongBench tasks, using a Mistral-7B-v0.1 pre-trained model with 7.25B parameters (Jiang et al., 2023). The three tasks chosen in LongBench are QuALITY (Pang et al., 2022), QMSum (Zhong et al., 2021), and NarrativeQA (Kočiskỳ et al., 2018).

QuALITY is a multiple-choice reading comprehension dataset over long articles (avg. ∼4,000 tokens), where each example presents an article with a 4-answer multiple choice question. The fine-tuned model must learn to select the correct option, with the cognitive bottleneck of *sequential evidence integration*. The model is judged on the top-1 accuracy, i.e., exact match of predicted option letter to gold.

QMSum is a query-focused summarization dataset of meeting transcripts (avg. ∼10,000 tokens per meeting, truncated to 4096 tokens). Given a query and a transcript, the fine-tuned model must learn to generate a paragraph-length summary addressing the query. Since meeting transcripts are inherently sequential, turn-taking, speaker contributions, and topic shifts follow a temporal order that is meaningful for summarization. We measure ROUGE-1, ROUGE-2, and ROUGE-L against reference summaries.

Finally, NarrativeQA is an open-ended question answering dataset over full books and movie scripts (avg. ∼50,000 tokens per document). For each document-question pair, the answer is a specific phrase or sentence from the document. The fine-tuned model must identify and generate the exact answer phrase from within the document context. We measure token-level F1 score (case-insensitive, following standard NarrativeQA evaluation).

All methods are iso-parametric, with HRM state dimension $d = 32$ and LoRA rank $r = 16$, both yielding ≈ 8.4M trainable parameters, roughly 0.116% of the total. Training protocol was identical across all methods: 5 epochs, lr= 5×10^{-4}, batch size=1, gradient accumulation=8 (effective batch=8), max input length= 4096, with AdamW optimizer. Evaluation uses the held-out 10% test split for QuALITY and QMSum, and the official test split for NarrativeQA.

We found that of the 3 tasks, HRM outperformed all: LoRA, DoRA, QLoRA, and AdaLoRA on two tasks – QuALITY and QMSum. While it drastically underperformed all methods on NarativeQA. HRM exceeded the best baseline (LoRA & AdaLoRA) on QuALITY on the accuracy metric by +34.8% (HRM acc. 47.43% to 35.18% for LoRA/AdaLoRA). HRM exceeded the best baseline (QLoRA) on QMSum on all 3 metrics: ROUGE-1, ROUGE-2, ROUGE-L. The ROUGE-1 metric was improved by +54.3% relative (HRM R-1 0.2531 to 0.1641% for QLoRA), by +73.59% relative against DoRA, and by +71.36% relative for both LoRA and AdaLoRA. However, it drastically underperformed on NarrativeQA, with an F1 score of 0.0391 against the best model (LoRA, F1=0.1592), under performing by a whole -75.4% (see Appendix .F for details).

Remark 7.1. Similar to 6.1, Mistral-7B backbone is not suited for SoTA performance on LongBench, as the goal is to compare adapter improvements across different types of tasks that involve long context reasoning. The is supported in our findings for the 3 tasks.

QuALITY requires the model to read a ∼4K-token article and answer a multiple-choice question, a prototypical sequential integration problem. Geva et al. (2021) establish that transformer MLP blocks function as key-value memories to perform content integration and associative recall, while attention performs positional retrieval. Fine-tuning with LoRA modifies the attention's Q, K, V projections, improving the model's ability to attend to relevant spans. However, for MCQ reasoning, the pre-trained model can already attend to relevant spans and the bottleneck is integrating evidence across multiple spans into a coherent

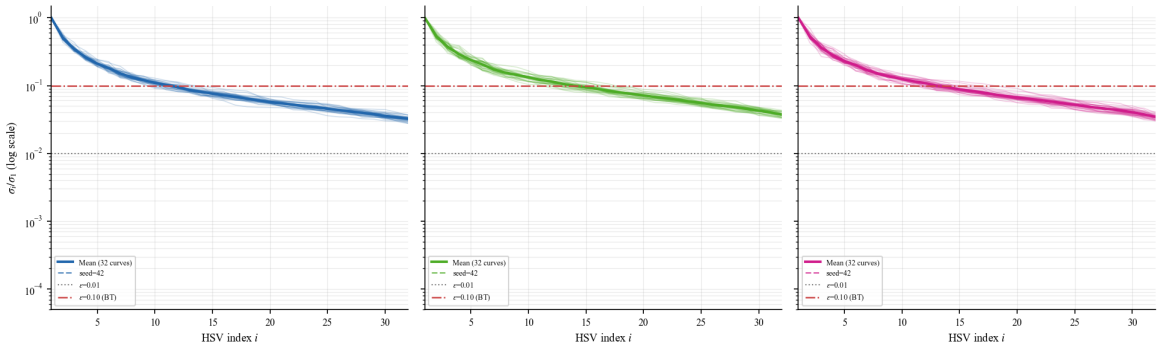

Figure 4. Mistral-7B HRM: Hankel Singular Value decay curves (left) QuALITY, (middle) QMSum, (right) VarrativeQA.

conclusion. The HRM adapter provides a recurrent SSM residual that maintains a running integration of the MLP's content representations across sequence positions. The relative improvement is therefore consistent with this mechanistic prediction.

QMSum requires generating a focused summary of a meeting transcript in response to a query. Due to similar reasoning as above, we can explain HRM's relative gains over the LoRA family for PEFT.

NarrativeQA on the other hand fails completely. Part of this was because of the hardware constraints of a dual NVIDIA GeForce RTX 4090 GPUs setup, with a total VRAM of 48GB. Due to this, the complete NarrativeQA context of 50,000 was not used at all, and truncated to 5,000 for training. As a result, all methods in comparison failed the benchmark, as the NarrativeQA task was not suitable for the hardware. As a result, HRM vastly underperformed all LoRA family on the task.

At the coarser threshold ε =0.10, the complexity ordering extends to LLM-scale tasks. From BT on trained Mistral-7B checkpoints (32 layers), $\hat{d}$(DFA, ε =0.01)=5 $<$ $\hat{d}$(QuALITY, ε =0.10) $\approx$11 ¡ $\hat{d}$(QMSum, ε =0.10)$\approx$13 $<$ $\hat{d}$(`enwiki8`, ε =0.01)=32 $\leq$ $\hat{d}$(MAESTRO, ε =0.01)=32. The HSV decay plot for the three tasks is shown in Fig. 4. Gate analysis reveals that HRM gates converge near zero despite non-zero initialization and explicit weight-decay exclusion: the gradient itself drives closure. Yet frozen-gate probing confirms that the trained SSM weights (A, B, C) retain 1180$\times$ larger contribution capacity at gate=0.1 This is shown in the final gate values learned for each layer, for each of the three tasks in Fig. 5.

8. Conclusion and Discussion

In this work, we proposed Hankel reduced order model (HRM) adapter for parameter efficient fine-tuning (PEFT). HRM adds a provably compressible recurrent temporal state to any frozen pre-trained transformer backbone, and unlike prior PEFT methods, the adapter is temporal causal. The model reduction is achieved via balanced truncation of the underlying linear system, where we utilize controllability and observability Grammians from control theory. This allows us to have a (tight) error bound making model reduction a certified compression, and not a heuristic. PEFT literature has inadvertently restricted itself to the class of zero-memory adapters. HRM results show that a gated SSM, inserted parallel to a frozen transformer's MLP blocks, can be trained efficiently, compressed with theoretical guarantees, and consistently outperforms the best static alternative on temporal tasks. We demonstrate that HRM outperforms LoRA, QLoRA, AdaLoRA, and DoRA on 6 different tasks, from three qualitatively different task families that share the requirement of causal state accumulation We also observe as a by-product that Hankel singular values and associated Grammians are a strong metric for the training task's memory requirements.

Open Problems & Next Steps Numerous immediate next steps emerge from the presented HRM works. 1. Extension of Hankel singular value-based balanced truncation to selective SSMs (Mamba-type, with input-dependent $\bar{A}_t$) benefits the proposed adapter. This would involve computationally tractable parameter-varying empirical Grammian computations that do not cause overhead larger than the intermediate FFT/IFFT calculations. 2. Currently the BT compression takes place after an initial phase's SSM adapter training. An enhancement would be to *adaptively allocate Hankel ranks for each layer*, i.e., $\hat{d}^{(l)}$ for layer l. This is analogous to AdaLoRA's rank allocation, but would be informed by per-layer $\sigma_i^{(l)}$ HSV spectra. 3. More complete benchmarking on LongBench of existing results is needed, for more tasks, across deeper context lengths. 4. Mixed injection of the SSM adapter needs to be ablated: attention injection site vs. MLP injection site, and more importantly, finding out task signatures suitable for each injection site for the adapter, and examine if simultaneous injection would benefit retrieval + integration tasks. 5. Investigate frozen gate training scenarios (e.g., `gate.requires_grad=False`) to

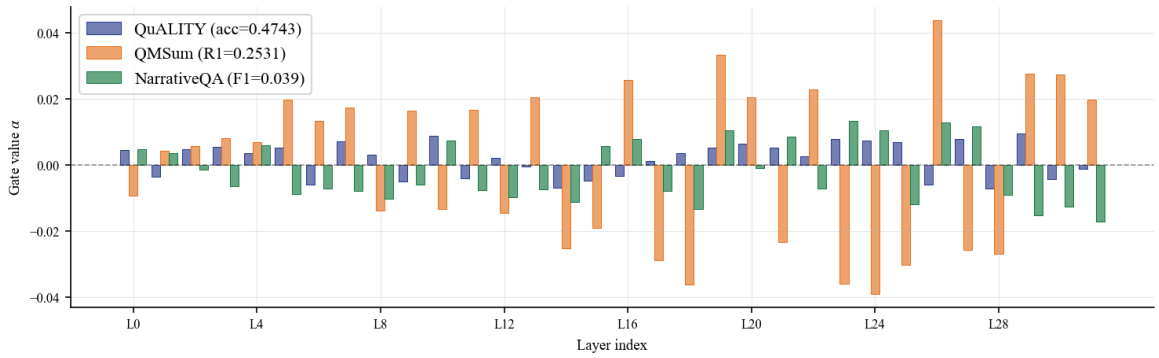

Figure 5. HRM gate values per layer for Mistral-7B, all three LongBench tasks (gate_init = 0.1).

enforce sustained HRM adapter contribution, and directly test whether larger training times translate to performance gains.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 7319–7328, 2021.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Corless, M. J. and Frazho, A. *Linear systems and control: an operator perspective*. CRC Press, 2003.

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

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Glover, K. All optimal hankel-norm approximations of linear multivariable systems and their L^∞-error bounds. *International journal of control*, 39(6):1115–1193, 1984.

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

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

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

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. *Advances in neural information processing systems*, 35: 35971–35983, 2022.

Hao, Y., Cao, Y., and Mou, L. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*, 2024.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

Hayou, S., Ghosh, N., and Yu, B. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In

International conference on machine learning, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Kočiskỳ, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Lall, S., Marsden, J. E., and Glavaški, S. Empirical model reduction of controlled nonlinear systems. *IFAC Proceedings Volumes*, 32(2):2598–2603, 1999.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965, 2022.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

Mahoney, M. Large text compression benchmark, 2011. *URL http://www. mattmahoney. net/dc/text. html*, 2013.

Moore, B. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE transactions on automatic control*, 26(1):17–32, 2003.

Pang, R. Y., Parrish, A., Joshi, N., Nangia, N., Phang, J., Chen, A., Padmakumar, V., Ma, J., Thompson, J., He, H., et al. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, 2022.

Park, J., Park, J., Xiong, Z., Lee, N., Cho, J., Oymak, S., Lee, K., and Papailiopoulos, D. Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*, 2024.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 46–54, 2020.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., et al. Slora: Scalable serving of thousands of lora adapters. *Proceedings of Machine Learning and Systems*, 6:296–311, 2024.

Zhang, M., Chen, H., Shen, C., Yang, Z., Ou, L., Yu, X., and Zhuang, B. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. 2023a.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023b.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Hassan, A., Celikyilmaz, A., Liu, Y., Qiu, X., et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, 2021.

A. Iso-parameter Comparison

In order to compare HRM and LoRA on an equal footing, we ensure that r and $\hat{d}$ are chosen such that $|P_{LoRA} - P_{HRM}| \leq 0.1\%$. Such an iso-parametric table to choose r and $\hat{d}$ is shown below. All experiments in the paper use all three tiers to demonstrate consistency, and conclusions are drawn only from the pattern across tiers rather than individual tier performance.

Table 1. Comparison of parameter counts between LoRA and HRM configurations.

LoRA r	$P_{\textbf{LoRA}}$	**HRM** $\hat{d}$	$P_{\textbf{HRM}}$	Δ **(%)**	Regime
8	16,384	16	16,156	$+0.81$	Low capacity
16	32,768	32	33,028	$+0.79$	Mid capacity
32	65,536	63	65,020	-0.79	High capacity

B. Parallel Scan for HRM

Since $\bar{A}$ is diagonal with entries $\bar{a}_1, \cdots, \bar{a}_d$, the impulse response factorizes: $g_k = C\bar{A}^k\bar{B} = C\text{diag}(\bar{a}_1^d, \cdots, \bar{a}_d^d)\bar{B}$. The ith column of $\bar{A}^k\bar{B} = \bar{a}_i^k cdot \bar{B}_i$, so the full convolution can be implemented as d scalar convolutions (one per state dimension), each with a geometric impulse response. This is the efficient form used in the code. On the other hand, the FFT forward pass computes for each state dimension i, $g_k^{(i)} = \bar{a}_i^k$ for $k = 0, \cdots, T-1$, being $\mathcal{O}(T)$ per dimension. Next, FFT($\tilde{g}^{(i)}$) and $\tilde{h}^{(i)}$ is $\mathcal{O}(T \log T)$ per dimension, giving a total of $\mathcal{O}(T \log T)$. The element-wise product $\odot$ and IFFT computations in (25) are $\mathcal{O}(dT \log T)$, and the final $\bar{B}, C$ projections are $\mathcal{O}(d \cdot d_{model} \cdot T)$. The net complexity for the FFT forward pass is therefore, $\mathcal{O}(d_{model} dT + dT \log T)$. For example, if the model order d, or $\hat{d}$ is 16, then $d \ll d_{model} = 128$, the next complexity if dominated by $\mathcal{O}(d_{model} \cdot d \cdot T)$.

Table 2. Epoch training times on MacBook Pro M2 (MPS backend). HRM-sequential is 10–14× slower than LoRA. HRM-FFT matches LoRA exactly at tested context lengths.

Context Length T	**LoRA Wall clock time/epoch (s)**	**HRM-seq. Wall clock time/epoch**	**HRM-seq. / LoRA**	**HRM-FFT Wall clock time/epoch**	**HRM-FFT / LoRA**
512	9	∼90	10×	9	1×
1024	70	∼980	14×	70	1×
2048	265	∼3710	14×	265	1×

Since FFT operations in `float32` introduce rounding errors of order 10^{-7}. We verify empirically compare sequential and FFT outputs across 100 random sequences. We find that the maximum absolute error is $< 5 \times 10^{-6}$, negligible for gradient computation, confirming the FFT equivalence is exact up to floating-point rounding (Fig. 6).

We compared the compute for with and without the FFT-based parallel scan over varying context lengths. Training at T=2048 with batch=32 required issuing 2048 Python-level dispatcher calls per layer per forward pass. At 4 layers and batch=32, this is 4 × 2048 × 32 = 262,144 sequential operations — completely saturating the Python-PyTorch overhead.

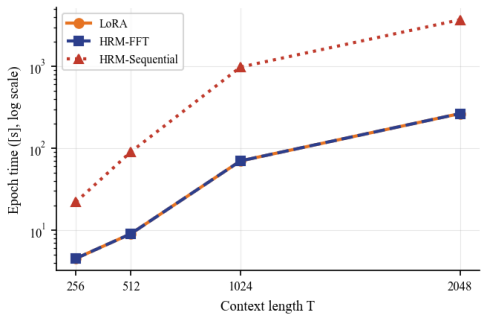

Figure 6. Maximum absolute error between FFT and sequential scan outputs over 100 random input sequences at varying T.

C. Is there a Case for SVD over HSVs?

A singular value decomposition to reduce model order seems like a natural alternative to balanced truncation. Several works apply this idea to LoRA matrices: FLORA (Hao et al., 2024), LoRA pruning (Zhang et al., 2023a). We argue that SVD is the wrong tool for a

dynamical system, and BT is the correct one. For static matrix order/rank reduction, SVD is a more natural choice. For instance, a state direction may have a large singular value in $\bar{B}$ (i.e., the input strongly drives that direction) yet be completely unobservable, i.e., producing zero output at channel corresponding to C. SVD of $\bar{B}$ would retain this direction as important, whereas its Hankel singular value (HSV) would be $\sigma_i = 0$, thereby getting discarded. Conversely, a direction with small $\bar{B}$ singular value might be the only one observable at C, and SVD would drop it while its HSV would preserves it.

SVD identifies directions that are large in individual matrices. HSVs identify directions that are simultaneously reachable and observable in the complete input-output system. Due to the causal input-output relation encoded by the Hankel operator, a singular value of $\bar{B}$ alone would not be able to drive model reduction. Only the Grammian product $W_c W_o$ gives the correct importance measure in this case, since accounts for the full dynamics. We empirically confirm this on the DFA task, where the largest singular value of $\bar{B}$ does not correspond to the dominant Hankel singular value σ_1. If we had truncated by SVD($\bar{B}$), we would have retained different state directions than BT and achieved a worse compression ratio. The BT result $\hat{d}$=6 with $< 0.3\%$ accuracy loss would likely not have been achievable by SVD of any single weight matrix.

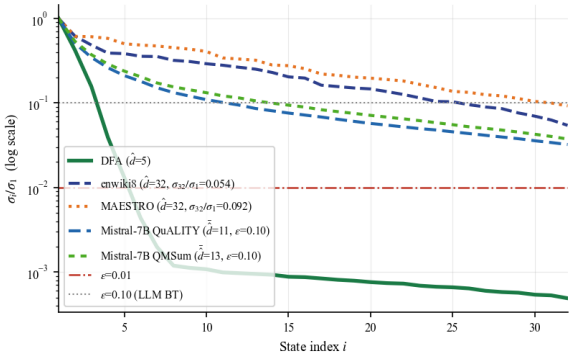

Figure 7. HSV decay for: DFA, `enwik8`, MAESTRO, QuALITY, and QMSum.

C.1. Further Insights from HSVs

Since the Hankel operator encodes the dynamical input-output relation as $\mathcal{H} : \{\text{past inputs}\} \to \{\text{future outputs}\}$, the HSV decay curve σ_i/σ_1 characterizes the task's temporal complexity. For instance, a steep decay curve signifies that most of the temporal complexity of the dynamics are contained in very few dynamical modes, therefore, $d \to \hat{d}$ HRM is highly compressible. Conversely, a gradual decay indicates a genuine high-dimensional memory requirement. To this end, *HSVs, and associated (empirical) Grammians are a strong metric for the training task's memory requirement*. Fig. 7 collects all available HSV decay results presented in this work.

D. `enwik8` Task

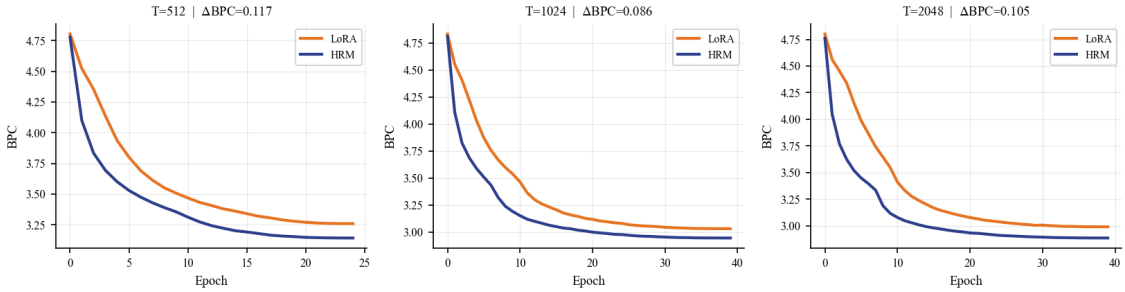

Figure 8. `enwiki8` BPC learning curves at Tier 2 (HRM $d = 32$ vs LoRA $r = 16$) for T $\in$ 512, 1024, 2048. The region between curves represents the BPC advantage of HRM over LoRA.

The HRM adapter's dominant state mode has a learned eigenvalue $\bar{a}_{\max}$ ($\bar{a}_{\max}$ ≈0.97–0.99 after training). The fraction of signal retained from a token k steps ago is $\bar{a}^k_{\max}$. At T=512, the adapter retains $\bar{a}^{256}_{\max} \approx 0.97^{256} \approx 0.0006$ of signal from the midpoint of the context window. This means that only the most recent ∼70 tokens contribute substantially (at 1% threshold). At T =2048, the same adapter covers proportionally more of the window — now $\bar{a}^{1024}_{\max} \approx 10^{-13}$, but $\bar{a}^{7}_{\max}0$ ≈0.11 still holds. The key insight is that the adapter's effective memory depth is fixed by the trained eigenvalues, but the fraction of context this covers grows with T. At T=2048, the 70-step memory covers 3.4% of context; at T=512, it covers 13.7%. The LoRA static ΔW covers none. As T grows, HRM's absolute reach stays roughly constant while LoRA's relative disadvantage grows.

Table 3. Comparison of BPC across different adapter capacities and context lengths.

Adapter Capacity	Context length	LoRA BPC	HRM BPC	ΔBPC
$r = 18, \hat{d} = 16$	512	3.3006	3.1902	0.1104
	1024	3.0271	2.9831	0.0440
	2048	3.0039	2.9026	0.1013
$r = 16, \hat{d} = 32$	512	3.2594	3.1424	0.1170
	1024	3.0326	2.9464	0.0862
	2048	2.9911	2.8859	0.1052
$r = 32, \hat{d} = 63$	512	3.1789	3.1470	0.0319
	1024	3.0186	2.9413	0.0773
	2048	2.9906	2.8789	0.1117

E. Parity Task

The parity task requires predicting, at each position t, the parity bit $p_t = \oplus_{k \le t} \sigma_k (\text{mod } 2)$ where $\sigma_k \in \{0, 1\}$. Unlike DFA state tracking (which requires tracking $k = 4$ states), parity has a minimal state of exactly 1 bit. An ideal adapter would learn $\hat{d}$=1. In practice, HRM converges to $\hat{d} \approx$ 6-7 (BT threshold ε=0.01), which is higher than expected, suggesting the model learns redundant but numerically stable representations of the parity state. The task serves as a lower bound: if HRM cannot outperform LoRA here, it is unlikely to help on any sequential task.

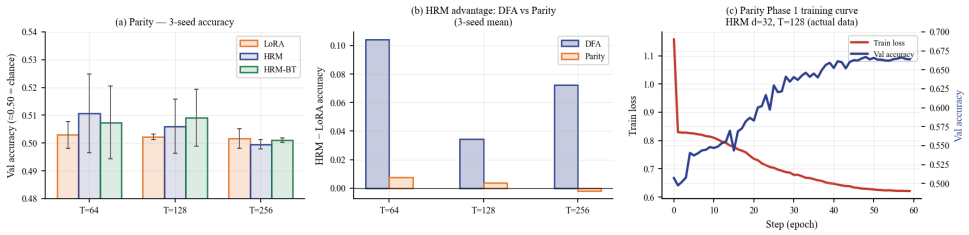

Figure 9. (left) Parity accuracy at medium model capacity, 3-seed mean ± std. All adapters near chance (0.50), parity is near-intractable for a small frozen backbone at $T = 256$. (middle) HRM advantage (HRM mean-LoRA mean) on DFA vs. parity, (right) training curve

.

Our observations (shown in Fig. 9) show that DFA exhibits a large, advantage with T while parity shows essentially zero HRM benefit. This contrast validates the memory hypothesis: HRM helps when multi-dimensional state is required, not when the task can be solved by single-bit counting.

F. LongBench Tasks

Table 4. Comparison of HRM against baselines on LongBench: QuALITY, QMSum, NarrativeQA

Method	QuALITY (Acc)	QMSum (R1/R2/R)	NarrativeQA (F1)
LoRA	0.3518	0.1477 / 0.0133 / 0.0925	0.1592
AdaLoRA	0.3518	0.1477 / 0.0133 / 0.0925	0.1571
DoRA	0.3478	0.1458 / 0.0133 / 0.0916	0.1569
QLoRA	0.3399	0.1641 / 0.0140 / 0.1038	0.1492
HRM	0.4743	**0.2531 / 0.0339 / 0.1180**	0.1492
HRM vs. best baseline	*+0.1225 (+34.8%)*	*+0.0890 R-1 (+54.3% vs LoRA)*	*-0.1201 (-75.4%)*

G. Ablations

BT Threshold ε The BT threshold ε controls the trade-off between compression ratio ($\hat{d}/d$) and accuracy loss. A smaller ε means fewer dimensions pruned, and a higher $\hat{d}$ implies a better accuracy but less compression. Conversely, a larger ε results in more aggressive pruning. We capture the ablation in the Figure below. Fig. 10 shows that $\hat{d}$ as a function of $\hat{d}$ for DFA at T $\in$ 64,128,256,512: $\hat{d}$ drops from 30–32 at $\hat{d}$=0.001 to 2–5 at $\hat{d}$=0.2. The elbow in the $\hat{d}$ vs $\hat{d}$ curve near $\hat{d}$=0.01 is the natural compression point, further pruning causes accuracy degradation (panel b). Default $\hat{d}$=0.01 is chosen at this elbow.

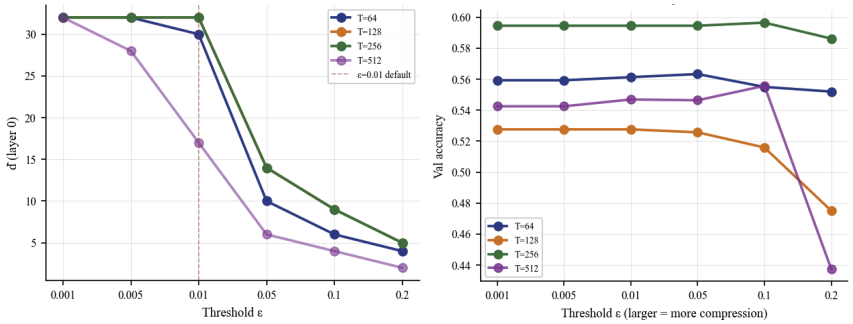

Figure 10. BT threshold ablation on DFA (left) DFA: $\hat{d}$ vs $\hat{d}$ threshold (layer 0), (right) DFA: accuracy vs ε threshold.

Rank vs. Accuracy To verify that HRM's advantage is not merely a parameter count artifact, we sweep LoRA rank $r \in \{4,8,16,32,64,128\}$ and HRM state_dim d $\in \{4,8,16,32,64\}$ on DFA at T=128 and T=256. From Fig. 11, HRM (blue) shows monotonically improving accuracy with state_dim on DFA. LoRA (orange) plateaus and even degrades at high rank (r =64,128), a sign of overfitting to position-level features. The HRM accuracy at d =32 (0.562) exceeds the best LoRA at any rank (0.522 at r =16), confirming the advantage is structural not parametric.

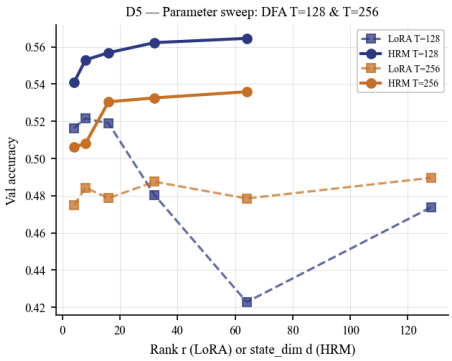

Figure 11. DFA T=128 and T=256: LoRA (rank) vs HRM (state_dim).

Data Efficiency HRM requires training to discover useful dynamics, followed by balanced truncation. We test whether HRM is less data-efficient than LoRA at small n_train. Fig. 12 shows that on DFA task, HRM requires n_train $\geq$ 2500 to match LoRA; below that, LoRA's simpler parameterization is more data-efficient. At n_train=10000 (paper default), HRM substantially outperforms LoRA. On parity task, both adapters are near-chance at all data sizes. As a result, the practical guidance from the ablation (for these tasks) is that HRM requires a minimum number of ($\sim$2000 in this case) training sequences to be competitive with LoRA on DFA-difficulty tasks.

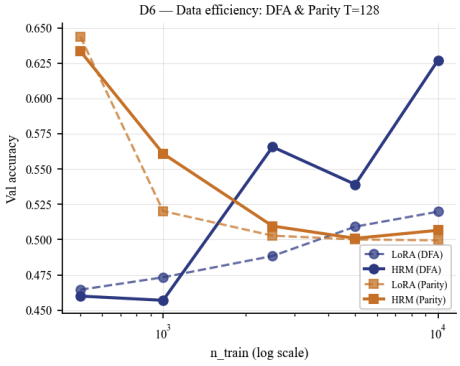

Figure 12. Data efficiency for DFA and Parity T=128: val accuracy vs. n_train.