

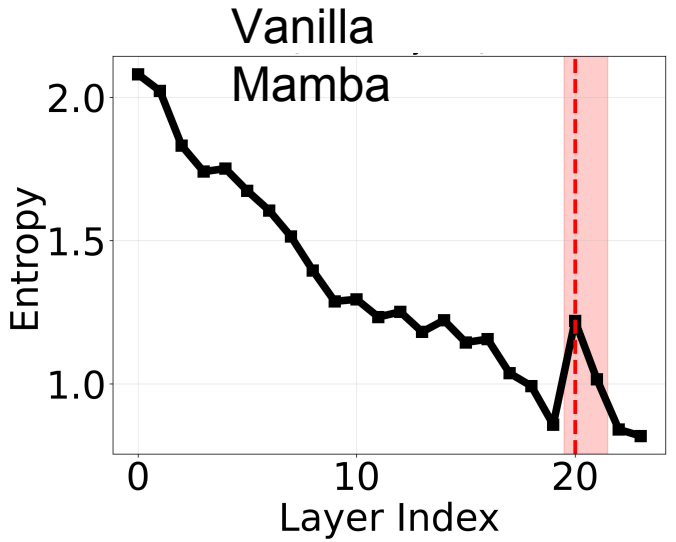
Interpreting and Steering State-Space Models via Activation Subspace Bottlenecks

Vamshi Sunku Mohan[1], Kaustubh Gupta[2], Aneesha Das[2], Chandan Singh[3]

[1]AppViewX, [2]Independent Researcher, [3]Microsoft

Motivation

- SSMs
  - Sequence models with recurrent hidden states
  - Lack explicit neuron/head structures found in Transformers. So, we use mechanistic interpretability
  - hard to interpret
  - weak on long-context
  - hard to steer
- Use mechanistic interpretability to understand how information flows
- Plot SPD entropy to study information flow across layers
- Information in Mamba is routed through a small number of dominant activation subspaces



Entropy spike at Layer 20
NIAH: 84%
QA: 66.7%
Pathfinder: 60%

Layer	Grad Sensitivity	KL-Divergence
19	0.811	1.0
20	0.072	813
21	0.790	1.0

Mechanistic Interpretability Analysis

Method	Description	Key Findings
Sparse Autoencoders (SAEs)	Reconstruct hidden activations using sparse latent representations to identify activation-level features and compact summaries of model activations	Identified sparse task-relevant features carrying most of the useful signal. SAE analysis revealed extreme sparsity and emergent specialized features responsible for key behaviors
Dictionary Learning	Applied on SAE latent representations to decompose activations into interpretable components and quantify sparsity, reconstruction error and feature utilization	Showed that sparsity, reconstruction error and feature usage occupy distinct latent dimensions, supporting functional specialization within representations
SPD (Stochastic Parameter Decomposition)	Decomposes parameters into sparsely used vectors and computes entropy, variance, effective rank, gradient sensitivity and post-ablation KL divergence	Identified a major Layer 20 information bottleneck characterized by high entropy, high variance and extreme KL divergence after ablation
Hypothesis Probing	Probe-based analysis examining whether task-relevant information is concentrated in sparse SAE-discovered features and circuits	SAE-based circuits showed strongest task alignment and predictive structure with structured positional selectivity
Polysemanticity Analysis	Measures feature interference, effective dimensionality and activation sparsity to analyze feature superposition	Found moderate superposition and low effective dimensionality, indicating substantial feature reuse across dimensions
Grokking Analysis	Analyzes emergent feature organization, induction behavior and transition dynamics during training	Observed sparse feature specialization, gradual learning dynamics and increasing induction strength during recursive integration
Causal Validation using Activation Patching	Copies, amplifies or ablates discovered circuit activations to evaluate causal necessity and sufficiency	SAE-based circuits showed strongest causal influence, supporting distributed rather than modular computation
Dynamic Universality Analysis	Measures cross-domain activation consistency and circuit generalization across linguistic and code tasks	Found strong universality across linguistic tasks but weaker consistency for code tasks, indicating specialized sequence-processing dynamics
Scaling Analysis	Compares feature specialization, N-gram subspaces and induction behavior across different model scales	Observed scale-dependent shifts in feature selectivity and hierarchical recursive buildup in Mamba models
APD (Attribution Path Decomposition)	Combined with SPD to decompose computation into operational phases and functional modules	Identified five operational phases including a critical Layer 20 bottleneck controlled by *dt_proj.bias*

Table A1. Summary of mechanistic interpretability methods, analyses and key findings

Layer 20 is the bottleneck

Bottleneck is scale invariant

dt_proj.bias causes bottleneck

Layer 20: *dt_proj.bias* = CHOKEPOINT

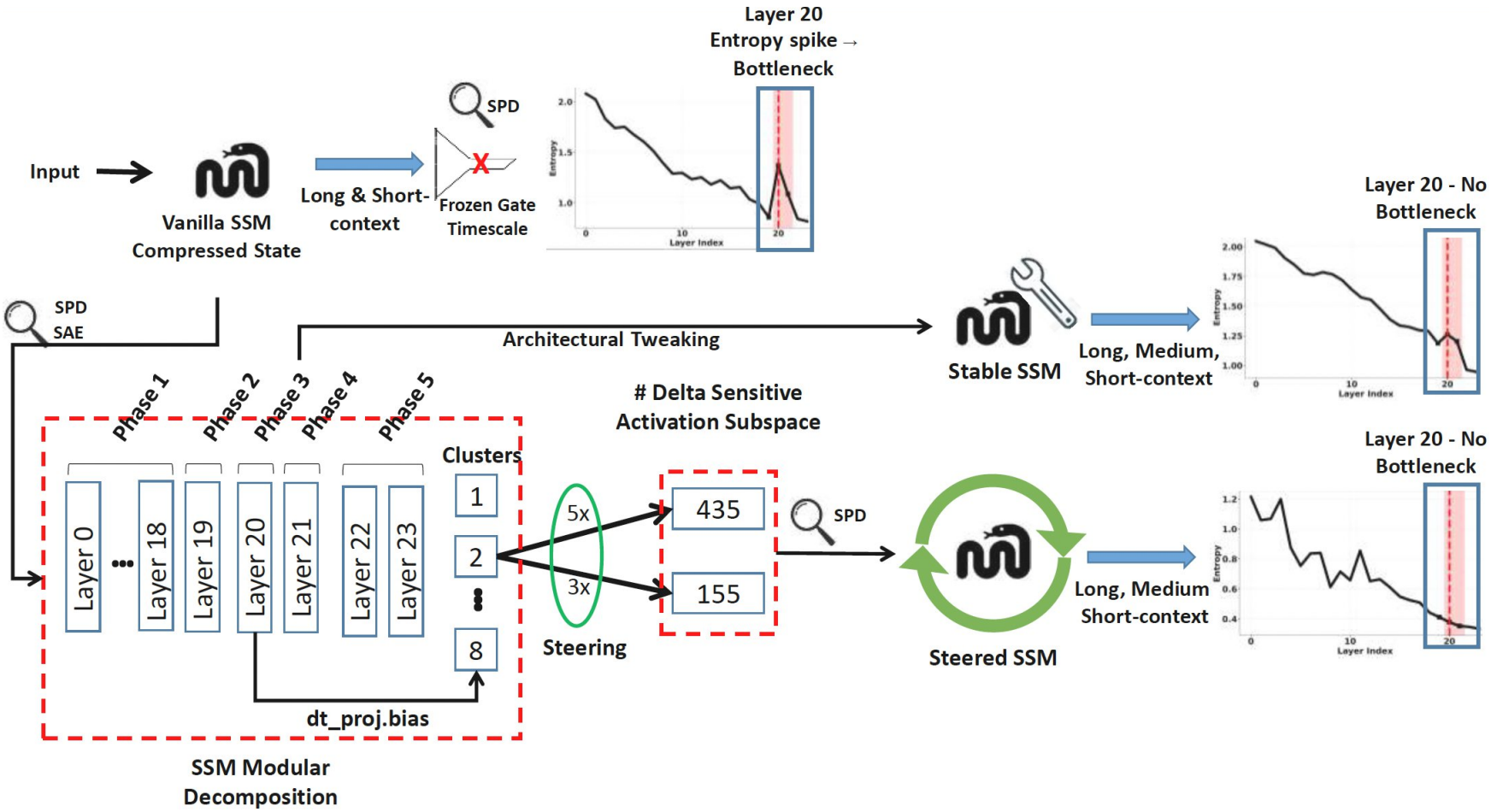
- SPD
  - Decomposes parameters into principal subspaces
  - Measures entropy across layers
    - High entropy = diverse information flow
    - Low entropy = bottleneck
- The smoking gun
  - Layer 20's *dt_proj.bias* - bottleneck parameter
- Why it matters mechanistically
  - Delta (Δ) gate in Mamba controls how much each token updates the hidden state
  - When *dt_proj.bias* collapses, all tokens are routed through the same narrow subspace ☾ Information diversity is lost.

Model	Total Layers	Bottleneck Layer	Depth Ratio (%)
Mamba-130M	24	20	83.3
Mamba-370M	48	40	83.3
Mamba-790M	48	41	85.4
Mamba-1.4B	48	43	89.6
Mamba-2.8B	64	53	82.8

Bottlenecks consistently emerge at **~85% of model depth**, suggesting a scale-invariant information compression phenomenon across Mamba architectures

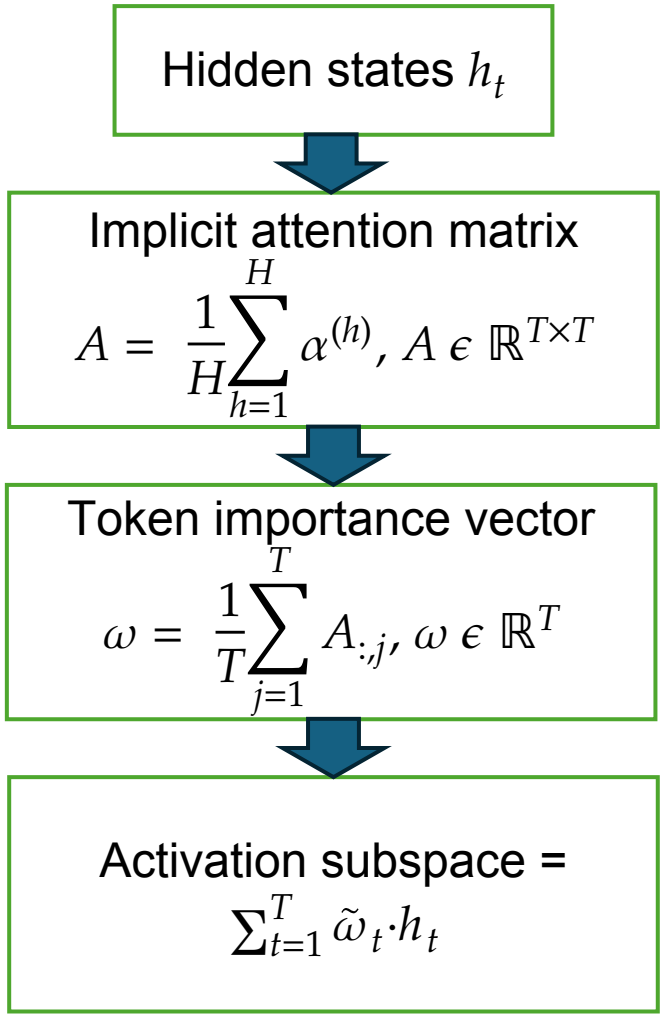
Core Analysis

- Delta-sensitive subspaces capture recurrent dynamics
- Two fixes:
  - Post-hoc steering
  - Architectural redesign (Stable-Mamba)



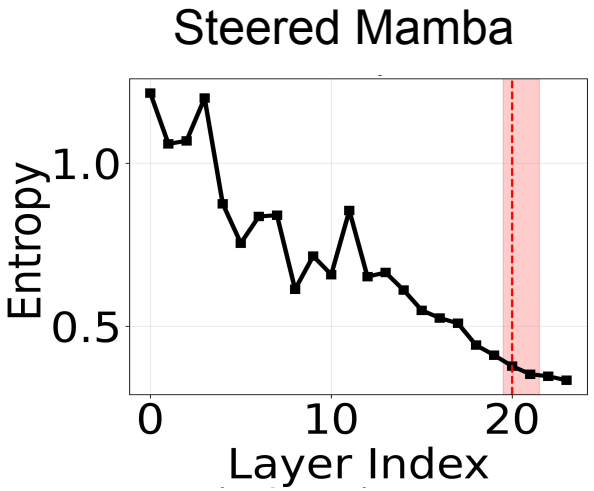
Delta-Sensitive Subspaces

- Mamba lacks explicit attention heads
- We extract *implicit attention matrix* from the recurrent dynamics
  - Showing which tokens each position attends
- Hidden states are projected onto attention-weighted directions
  - Subspaces with high variance (i.e., sensitive to changes in Δ) ☾ Active routing channels
- Delta-sensitive: High variance under recurrent state updates (Δ)
  - Used as steering targets ☾ Influences output
  - Localizes bottlenecks



Steering Intervention

- Steering redistributes information flow at inference time
- No task-specific tuning



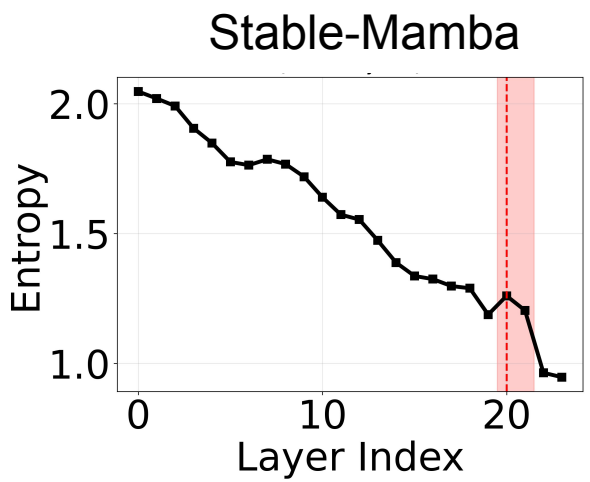
NIAH: 92.00%
QA: 76.0%
Pathfinder: 74.0%

Algorithm 1 Post-hoc Steering of Activation Subspace Bottlenecks on Mamba-130M

Compute SPD statistics for all layers
Identify bottleneck layers using entropy spikes and post-ablation KL divergence
for each bottleneck layer **do**
    Construct attention-style interaction matrix $A \in \mathbb{R}^{T \times T}$ from the recurrent `mixer.ssm` block
    Compute activation subspaces
    Record activation-subspace values across a large text corpus
    Identify Delta-sensitive subspaces exhibiting high variance across inputs
end for
Select layer with largest spike (Layer 20) as the primary bottleneck layer
for each causally selected Delta-sensitive subspace in Layer 20 **do**
    Ablate the subspace and measure the drop in next-token prediction accuracy
end for
Retain 590 Delta-sensitive subspaces for steering: 435 high-impact subspaces with performance drop >2% and 155 lower-impact subspaces with performance drop in (-2%, 2%]
for steering factor $s \in [0.1, 100]$ **do**
    Amplify all 590 subspaces by factor s
    Apply steering using forward hooks on $h_t \in \mathbb{R}^{B \times T \times d}$
    Measure change in next-token prediction accuracy
end for
Select the two best steering factors: $5.0\times$ with $+1.5\%$ accuracy improvement and $2.0\times$ with $+0.4\%$ accuracy improvement
Amplify the 435 high-impact subspaces by $5.0\times$
Amplify the 155 lower-impact subspaces by $2.0\times$
Evaluate the steered model on validation tasks

Stable-Mamba

- Adds 256 extra parameters
- Negligible inference overhead
- Entropy spike reduced
- Smoother information flow

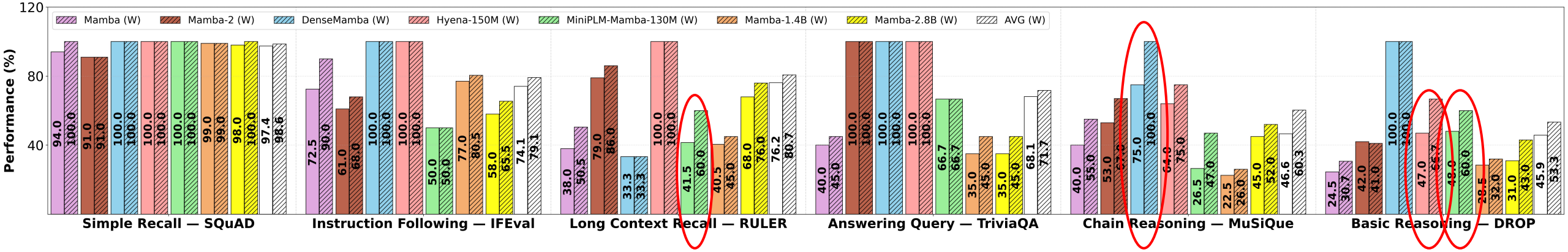


NIAH: 90.00%
QA: 82.0%
Pathfinder: 95.0%

Table E1. Component-wise architectural modifications introduced in Stable-Mamba and their impact on performance and stability

Component	Mamba	Stable Mamba	Performance Improvement in Tweaked Mamba
State transition	$h_t = \bar{A}h_{t-1} + \bar{B}x_t$ (single time-scale)	$h_t^{(k)} = \bar{A}_k h_{t-1}^{(k)} + \bar{B}_k(g \odot x_t)$ (3 time-scales: slow, medium, fast)	1) +23% perplexity improvement on long sequences (>1024 tokens) 2) +15% accuracy on multi-scale reasoning tasks 3) Entropy variance reduced by 31%
Tweaked Mamba	$y_t = Ch_t + Dx_t$ (linear)	$y_t = \sum_k w_k(C_k h_t^{(k)} + D_k x_t)$ (weighted ensemble)	1) Adaptive temporal resolution via learned weighting 2) Top features show 0.16+ importance vs. 0.02 in Mamba
Tweaked Mamba	Not present	$h_{\text{global}} = \text{SparseAttn}(h_{\text{local}})$ output $= \alpha h_{\text{global}} + (1-\alpha)h_{\text{local}}$	1) 22% reduction in bottleneck entropy (1.21 $\rightarrow$ 0.94) 2) Smoother information flow in layers 19–21 3) $\sim$0.4$\times$ cost of full Transformer attention
Gating	Uniform processing	$g = \sum_i \alpha_i \sigma(W_i \cdot \text{LN}(x))$ (ensemble gates, $n = 3$)	1) +683% feature usage efficiency (1.88% $\rightarrow$ 14.7%) 2) Sparsity reduced from 98.1% to 85.3%
Compression	Not present	$c = 0.5 + 0.5 \cdot \sigma(\text{MLP}(\bar{x}))$	1) Adaptive signal preservation with reduced noise 2) Effective compression strength in range 0.2-0.3
Gradient control	Standard backpropagation	$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \text{out}} \cdot \lambda_{\text{comp}} \cdot \frac{\partial \text{out}}{\partial \theta}$ (5 learnable scales)	1) 46% reduction in gradient instability (CoV: 17.3% $\rightarrow$ 9.4%) 2) Stable convergence without manual tuning
Residual	output $= y_t + x$	output $= (y_t + \lambda_{\text{res}} x)\lambda_{\text{global}}$	1) Improved gradient flow in deep networks 2) Learned residual scale 0.9 ± 0.1

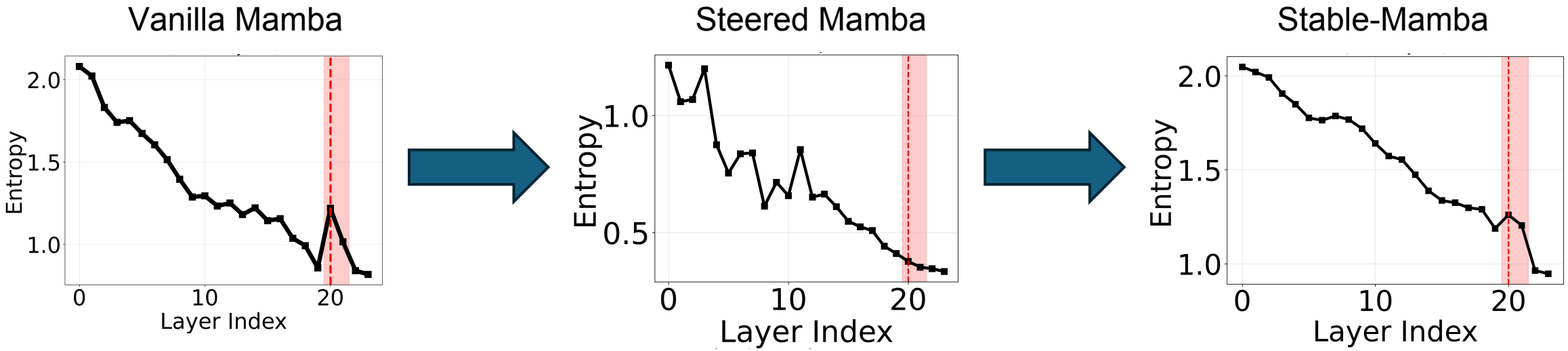
Results Across Models



- Steering improves:
  - Instruction following
  - Long-context retrieval
  - Chain reasoning
  - QA
- ~8.27% ↑ overall Next-token prediction accuracy
- ~23% ↑ Next-token prediction accuracy (excluding ceiling/floor cases)

- Steering transfers across architectures without task-specific tuning
- Bottleneck steering consistently improves models with concentrated entropy routing
- *"Steering = inference-time only; Stable-Mamba requires retraining"*

Conclusion



- Three takeaways:
  - Mamba has interpretable bottlenecks (Layer 20, *dt_proj.bias*)
  - Steering them improves performance without retraining
  - Interpretability insights directly guided better architecture

- ***Interpretability is not just analysis — it can directly improve model behavior***