From Markov to Laplace:
How Mamba In-Context Learns Markov Chains

Marco Bondaschi, Nived Rajaraman, Xiuying Wei, Kannan Ramchandran,
Razvan Pascanu, Caglar Gulcehre, Michael Gastpar, Ashok Vardhan Makkuva

EPFL · Berkeley University of California · DeepMind

State Space Models (SSMs)

	Transformers	Mamba
• Training:	Quadratic in T	Linear in T
• Inference:	Linear in T	Constant in T

In-context learning (ICL) drives transformers' success

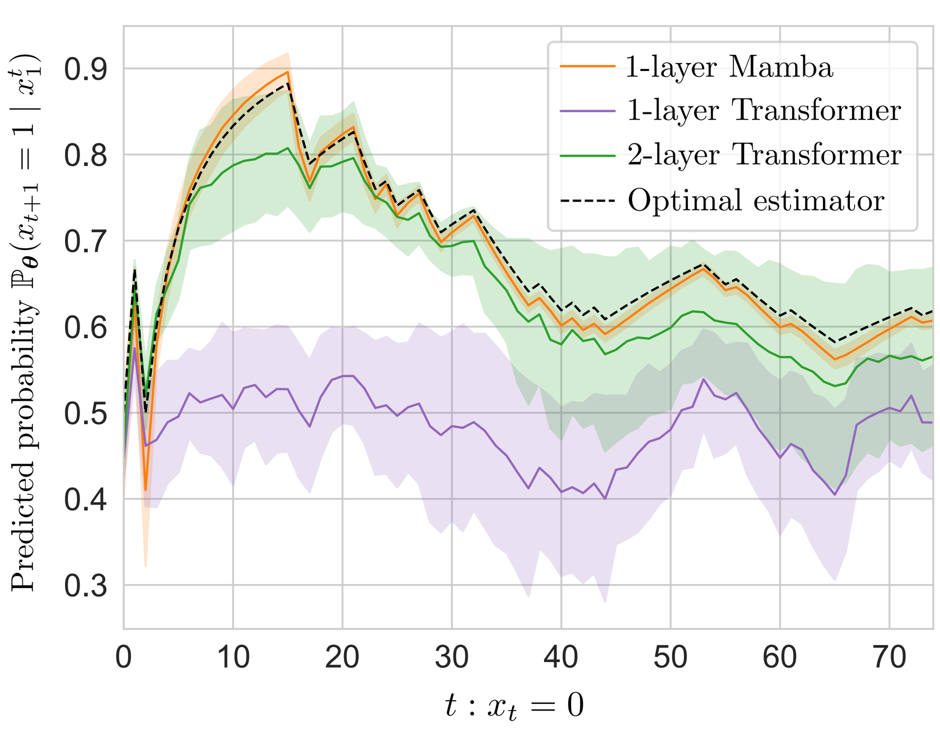
How can we study Mamba's ICL capabilities?

Mamba learns Laplacian smoothing

- Optimal predictor: **Laplacian smoothing**

$$\arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \mathbb{P}^*(x_{t+1}=1 \mid x_1^t) = \mathbb{E}_{P|x_1^t}\left[\mathbb{P}(x_{t+1}=1 \mid x_1^t)\right] = \frac{n_1+\beta}{n+2\beta}$$

It counts past transitions $x_{i-k}^{i-1} \to x_i$ and computes smoothed frequencies



Mamba learns the optimal predictor with *just one layer!*

Can we mathematically explain how?

Main theorems

Theorem 1 (*Representation power of Mamba*)

One-layer Mamba can learn the optimal Laplacian smoothing predictor for first-order Markov chains, provided that:

- State and hidden dimensions are at least 2: $N, d \geq 2$
- Convolution window is at least 2: $w \geq 2$

Key insights:

- Selectivity retains all the past: $a_t \approx 1$
- Convolution encodes transitions $x_{i-k}^{i-1} \to x_i$
- Hidden state H_t stores counts of past transitions

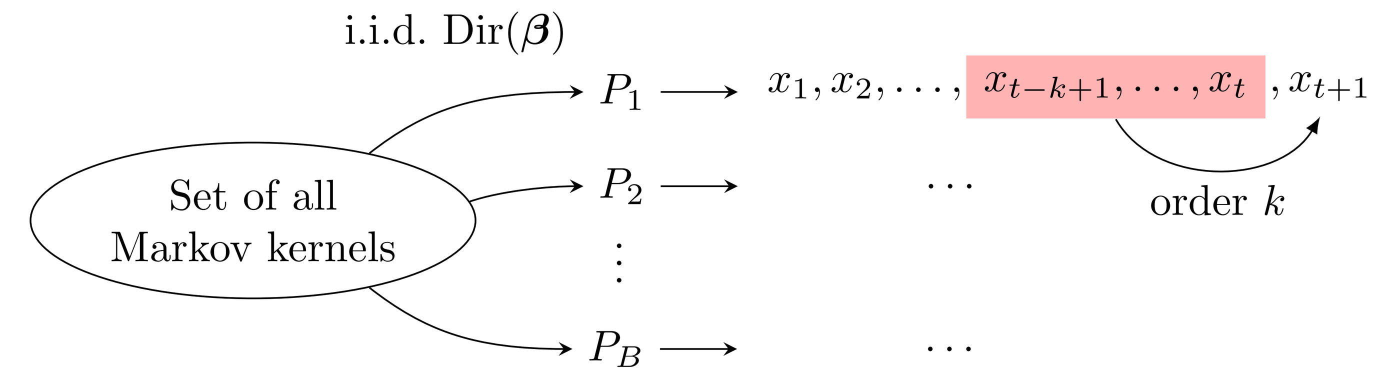
Theorem 2 (*Lower bound on the state dimension*)

One-layer Mamba cannot learn the optimal Laplacian smoothing predictor for any k-th order Markov chains, unless the state dimension is larger than 2^k:

$$N \geq C \cdot 2^k$$

In-context learning with Markov chains

- Data:



- Mamba-based LLM:

$$x_t \xrightarrow{\text{Embedding}} \boldsymbol{x}_t \xrightarrow{\text{Mamba}} \boldsymbol{u}_t \xrightarrow{\text{MLP}} \boldsymbol{v}_t \xrightarrow{\text{Linear}} \text{logit}_t \xrightarrow{\text{Prediction}} \mathbb{P}_{\boldsymbol{\theta}}(x_{t+1} \mid x_1^{t-1})$$

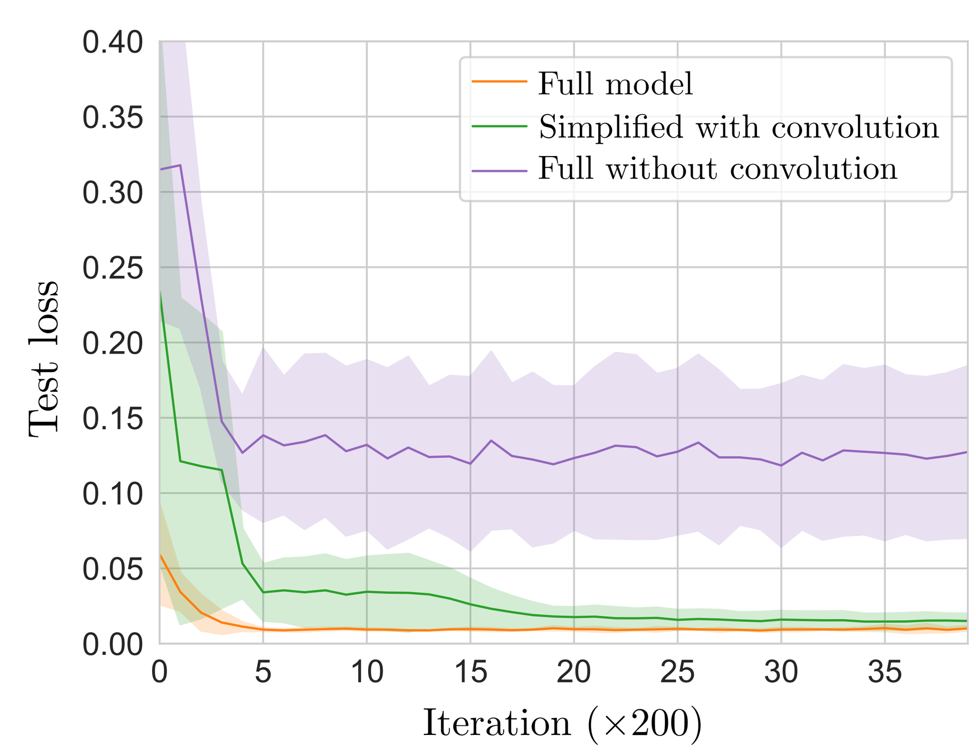
$$H_t = a_t\, H_{t-1} + \tilde{\boldsymbol{x}}_t \boldsymbol{b}_t^T$$
$$\boldsymbol{y}_t = H_t \boldsymbol{c}_t$$
$$\boldsymbol{z}_t = \boldsymbol{y}_t \odot \text{ReLU}(W_z\, \boldsymbol{x}_t)$$
$$\boldsymbol{o}_t = W_o\, \boldsymbol{z}_t$$

$$\tilde{\boldsymbol{x}}_t = \text{ReLU}(\text{conv}_X(W_X\, \boldsymbol{x}_{t-w+1}^t)) \cdot \Delta_t$$
$$\boldsymbol{b}_t = \text{ReLU}(\text{conv}_B(W_B\, \boldsymbol{x}_{t-w+1}^t))$$
$$\boldsymbol{c}_t = \text{ReLU}(\text{conv}_C(W_C\, \boldsymbol{x}_{t-w+1}^t))$$

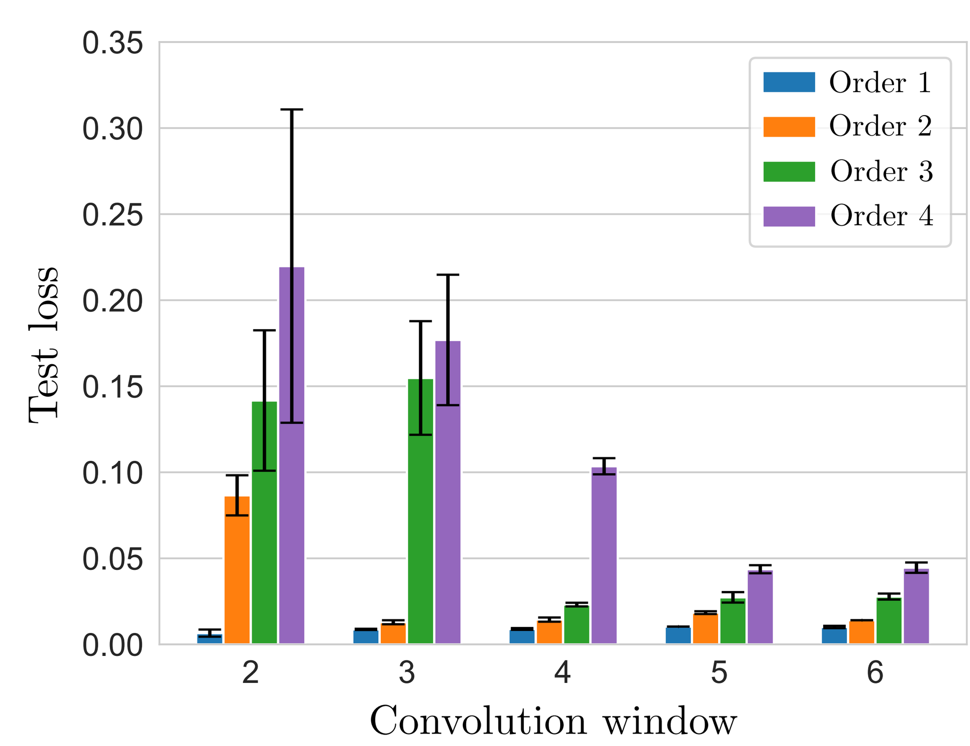
- Cross-entropy loss:

$$L(\boldsymbol{\theta}) = -\frac{1}{T}\sum_t \mathbb{E}\left[x_{t+1} \cdot \log \mathbb{P}_{\boldsymbol{\theta}}(x_{t+1}=1 \mid x_1^t) + (1-x_{t+1}) \cdot \log \mathbb{P}_{\boldsymbol{\theta}}(x_{t+1}=0 \mid x_1^t)\right]$$

Convolution is the key



Mamba needs convolution to learn the optimal predictor



Convolution window must be larger than the Markov order:

$$w \geq k+1$$

Beyond Markov

- Natural language: WikiText-103

Model	# Params. (M)	Perplexity (↓)
Mamba-2 (w/o conv)	14.53	30.68
Mamba-2 (w/conv)	14.54	**27.55**
Transformer (w/o conv)	14.46	29.28
Transformer (w/ conv)	14.46	**28.67**

Benefit of convolution goes beyond Markov data!

Find the full paper on arXiv: