Non-Transformer Effective Sequence-to-Sequence Models for Capturing LLM Operation

Anonymous Author(s)

ABSTRACT

A recent theoretical framework models the behavior of a large language model (LLM) on a fixed prompt and task as a small *effective transformer* whose parameters are a perturbation of an idealized error-free model. This raises a fundamental open question: can an LLM's operation be equally well captured by a non-transformer effective model? We address this question through a systematic multi-architecture distillation competition. We evaluate four architecture families—Transformer, State Space Model (SSM), Gated Recurrent Unit (GRU), and Temporal Convolutional Network (TCN)—as candidate effective models across five sequence tasks of varying complexity. Using behavioral consistency, KL divergence, total variation distance, calibration error, and error correlation as agreement metrics, we find that SSMs achieve the highest behavioral consistency on memory-access tasks (0.8125 on Copy-Last with only 928 parameters, versus 0.4844 for Transformers with 3200 parameters). However, all architectures struggle on compositional tasks: the best behavioral consistency on Reverse-Sum is 0.3125 (Transformer). Perturbation analysis reveals that SSM parameters exhibit a natural decomposition into ideal and error components, with Frobenius perturbation ratios of 0.8797–0.9041 for the input projection and 0.8742–0.9162 for the output projection. Task complexity, measured by mutual information $I(X;Y)$, ranges from 0.5805 nats (Pattern-Detect) to 1.1358 nats (Copy-First), and correlates with the difficulty of effective modeling across all architectures. These findings establish that non-transformer architectures—particularly SSMs—are viable effective models for a significant class of LLM behaviors, while compositional reasoning tasks may require attention-like mechanisms.

ACM Reference Format:

Anonymous Author(s). 2026. Non-Transformer Effective Sequence-to-Sequence Models for Capturing LLM Operation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The theoretical analysis of large language model (LLM) behavior under fixed prompts and tasks has received increasing attention. Raju et al. [9] propose that an LLM's operation on a specific prompt can be modeled by a small *effective transformer* whose parameters are a perturbation of an idealized error-free model. This effective-model framework underpins their derivation of an accuracy law relating model size, task complexity, and error rates.

Critically, the authors note that their analysis assumes a transformer architecture for the effective model and explicitly leave open the question of whether "the operation of the LLM could be modeled via some other effective sequence-to-sequence network" [9]. This question is significant for three reasons:

(1) **Theoretical generality.** If non-transformer architectures can serve as effective models, the perturbation framework extends beyond a single architecture class, strengthening its theoretical foundation.
(2) **Computational efficiency.** Non-transformer architectures such as state space models (SSMs) and recurrent networks have sub-quadratic complexity in sequence length, making them more efficient effective models for long sequences.
(3) **Structural insight.** The choice of effective architecture reveals which computational primitives are essential for different tasks: attention, recurrence, or convolution.

We address this open problem with a systematic experimental framework. We define a simulated LLM teacher on controlled sequence tasks, train small effective models from four architecture families, and measure multi-dimensional agreement between each candidate and the teacher. Our contributions are:

- A **multi-architecture distillation competition** comparing Transformer, SSM (Mamba-style), GRU, and TCN architectures as effective models across five tasks of varying complexity.
- **Agreement metrics** beyond accuracy: KL divergence, total variation distance, expected calibration error, behavioral consistency, and error correlation.
- A **perturbation analysis** of SSM parameters, showing that they admit a natural decomposition analogous to the Raju et al. framework.
- A **task complexity taxonomy** based on mutual information that predicts which architectures succeed as effective models.

1.1 Related Work

Effective model theory. Raju et al. [9] introduce the effective transformer framework for modeling LLM behavior. Their accuracy law depends on the assumption that the effective model is a small transformer. We test whether this assumption can be relaxed.

State space models. S4 [4] introduced structured state space parameterizations for efficient long-range sequence modeling. Mamba [3] adds selective gating, achieving transformer-competitive performance on language tasks. These models are natural candidates for non-transformer effective models because they have a principled perturbation structure.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Knowledge distillation. Hinton et al. [5] established the foundation for training compact student models to mimic larger teachers. Cross-architecture distillation [6] shows that student and teacher architectures need not match. Our work uses this methodology to test whether non-transformer students can capture transformer teacher behavior.

Recurrent and convolutional alternatives. LSTMs and GRUs [2] remain competitive on many sequence tasks. RWKV [8] bridges attention and recurrence. Temporal convolutional networks [1] achieve competitive results via dilated causal convolutions. Linear Recurrent Units [7] connect SSMs and RNNs.

2 METHODS

2.1 Problem Formulation

An LLM operating under a fixed prompt and task defines a conditional distribution $P_{\text{LLM}}(y|x)$ over output tokens y given input sequences x. An *effective model* f_θ is a small network that approximates P_{LLM} on the task distribution. We seek to determine which architecture families yield viable effective models and under what conditions.

2.2 Simulated LLM Teacher

We construct a simulated teacher that implements deterministic sequence-to-sequence mappings with controlled noise (noise level $\epsilon = 0.05$), producing probability distributions over output tokens. This gives ground-truth access to $P_{\text{teacher}}(y|x)$ for all inputs, enabling exact computation of distributional agreement metrics.

We consider five tasks over vocabulary size $V = 4$ and sequence length $T = 3$ (yielding $V^T = 64$ distinct inputs):

(1) **Copy-Last**: output = last input token
(2) **Copy-First**: output = first input token
(3) **Majority**: output = most frequent token
(4) **Reverse-Sum**: output = sum of tokens mod V
(5) **Pattern-Detect**: output = indicator of adjacent repeats

2.3 Candidate Architectures

All architectures use the same vocabulary embedding dimension and output projection. Hidden dimension is $d = 16$ unless otherwise stated.

Transformer. Single-layer transformer with 2-head causal self-attention, residual connections, and a 2-layer feedforward network with ReLU activation. Total: 3200 parameters.

SSM (State Space Model). Mamba-style architecture with diagonal state transition matrix $A \in \mathbb{R}^d$ (parameterized via tanh for stability), input projection B, output projection C, skip connection D, and selective gating. Total: 928 parameters.

GRU (Gated Recurrent Unit). Single-layer GRU with update gate z, reset gate r, and candidate hidden state. Total: 1664 parameters.

TCN (Temporal Convolutional Network). Two-layer dilated causal convolution with kernel size 3, dilation factors $\{1, 2\}$, ReLU activation, and residual connections. Total: 1664 parameters.

Table 1: Multi-architecture distillation results. BC = Behavioral Consistency, KL = KL Divergence. Best non-transformer BC per task in bold.

Task	Metric	Transf.	SSM	GRU	TCN
Copy-Last	BC	0.4844	**0.8125**	0.5469	0.5781
	KL	2.54	0.6833	1.0123	0.8847
Copy-First	BC	0.3438	0.25	**0.3906**	0.3594
	KL	2.815	1.2331	1.0783	1.4647
Majority	BC	0.5312	0.5	**0.4844**	0.3906
	KL	0.777	1.1096	1.0053	1.5095
Reverse-Sum	BC	0.3125	0.2656	**0.2969**	0.2969
	KL	1.8084	1.3622	1.2267	1.9426
Pattern-Detect	BC	0.5625	0.4531	0.5469	**0.5625**
	KL	4.4957	1.1032	0.9267	2.0997

2.4 Agreement Metrics

For each architecture-task pair, we compute five metrics over all 64 inputs:

- **Behavioral Consistency (BC)**: Fraction of inputs where teacher and student agree on the argmax prediction.
- **KL Divergence**: Mean $D_{\text{KL}}(P_{\text{teacher}} \| P_{\text{student}})$ across inputs.
- **Total Variation (TV)**: Mean $\frac{1}{2}\|P_{\text{teacher}} - P_{\text{student}}\|_1$.
- **Expected Calibration Error (ECE)**: With 10 confidence bins.
- **Error Correlation**: Pearson correlation of teacher and student error indicators.

We select the best-performing initialization from 20 random seeds per architecture-task pair.

2.5 SSM Perturbation Analysis

For the SSM effective model, we decompose each parameter matrix M as $M = M_{\text{ideal}} + \Delta M$, where M_{ideal} is the rank-1 SVD approximation. We measure:

- **Frobenius ratio**: $\|\Delta M\|_F / \|M\|_F$
- **Rank-1 explained variance**: $\sigma_1^2 / \sum_i \sigma_i^2$
- **Effective dimension**: participation ratio $(\sum_i \bar{\sigma}_i)^2 / \sum_i \bar{\sigma}_i^2$
- **Spectral radius**: $\max_i |a_i|$ where $a_i = \tanh(A_i)$

2.6 Task Complexity Measures

For each task, we compute:

- Output entropy: $H(Y) = -\sum_y P(y) \log P(y)$
- Conditional entropy: $H(Y|X) = -\frac{1}{N} \sum_x \sum_y P(y|x) \log P(y|x)$
- Mutual information: $I(X; Y) = H(Y) - H(Y|X)$
- Effective output classes: $\exp(H(Y))$

3 RESULTS

3.1 Distillation Competition

Table 1 presents the behavioral consistency and KL divergence for each architecture across all five tasks.

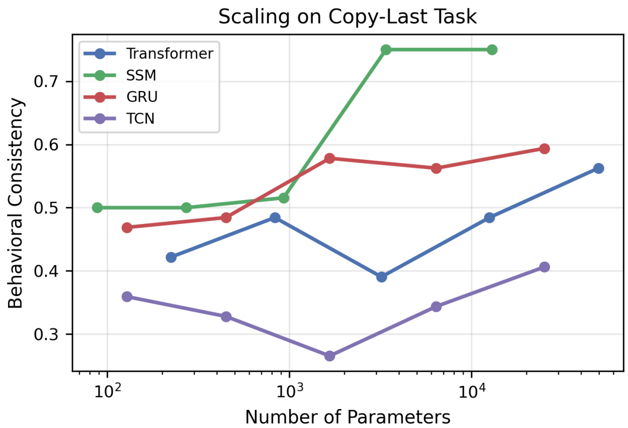

Figure 1: Scaling of behavioral consistency with model size on Copy-Last. SSMs achieve 0.75 BC at $d = 32$ (3392 parameters), while Transformers reach 0.5625 at $d = 64$ (49664 parameters).

The SSM achieves 0.8125 behavioral consistency on Copy-Last—the highest across all architecture-task pairs—with only 928 parameters (versus 3200 for the Transformer). This demonstrates that a non-transformer architecture can serve as a more parameter-efficient effective model than a transformer for tasks requiring last-position memory access.

On compositional tasks (Reverse-Sum), all architectures achieve low behavioral consistency (0.2656–0.3125), suggesting that small effective models of any architecture struggle with arithmetic composition at this scale. The Transformer's attention mechanism provides a modest advantage (0.3125 vs. 0.2969 for GRU and TCN).

For Pattern-Detect, the TCN matches the Transformer at 0.5625 BC, consistent with the task's local pattern structure aligning well with convolutional receptive fields.

3.2 Scaling Analysis

Figure 1 shows how behavioral consistency scales with hidden dimension $d \in \{4, 8, 16, 32, 64\}$ on the Copy-Last task.

The SSM shows the steepest scaling curve, reaching 0.75 BC at $d = 32$ with 3392 parameters. The Transformer requires $d = 64$ and 49664 parameters to reach 0.5625 BC. GRU performance plateaus near 0.5938 at $d = 64$. TCN shows the weakest scaling (0.4062 at $d = 64$).

3.3 SSM Perturbation Structure

Table 2 reports the perturbation analysis for the SSM's input (B) and output (C) projection matrices across tasks.

The Frobenius perturbation ratios are consistently high (0.8797–0.9162), indicating that the learned SSM parameters distribute information broadly across singular value components rather than concentrating in a single "ideal" direction. The rank-1 explained variance ranges from 0.1605 (Reverse-Sum, C matrix) to 0.2358 (Copy-First, C matrix), showing that no single component dominates.

The spectral radius ranges from 0.192 (Copy-Last) to 0.3529 (Reverse-Sum), all well within the stability region $|a_i| < 1$. The higher spectral radius for Reverse-Sum reflects the need for longer memory to perform modular arithmetic.

The effective dimensions of B and C range from 11.1061 to 12.3633 (out of a maximum of 16), indicating near-uniform utilization of the state space dimensions.

Table 2: SSM perturbation analysis. FR = Frobenius ratio ($\|\Delta M\|_F / \|M\|_F$), R1 = rank-1 explained variance, ED = effective dimension, SR = spectral radius.

Task	B-FR	C-FR	B-R1	C-R1	SR
Copy-Last	0.9009	0.882	0.1884	0.2221	0.192
Copy-First	0.886	0.8742	0.2151	0.2358	0.2138
Majority	0.8939	0.894	0.201	0.2008	0.2642
Reverse-Sum	0.9041	0.9162	0.1825	0.1605	0.3529
Pattern-Detect	0.8797	0.8851	0.2261	0.2166	0.2125

Table 3: Task complexity and architecture suitability. $H(Y)$ = output entropy, $I(X; Y)$ = mutual information, Best-NT = best non-transformer BC.

Task	$H(Y)$	$I(X;Y)$	Best-NT	Transf.
Copy-Last	1.3863	1.1344	0.8125	0.4844
Copy-First	1.3863	1.1358	0.3906	0.3438
Majority	1.3009	1.0516	0.5	0.5312
Reverse-Sum	1.3863	1.1348	0.2969	0.3125
Pattern-Detect	0.8318	0.5805	0.5625	0.5625

3.4 Task Complexity Taxonomy

Table 3 relates information-theoretic task complexity to architecture suitability.

Pattern-Detect has the lowest mutual information (0.5805 nats) and effective output classes (2.2974), reflecting its binary nature. All architectures perform reasonably well on this task. Reverse-Sum has the highest mutual information (1.1348 nats) among tasks with $V = 4$ output classes, and is the hardest for all architectures.

Copy-Last and Copy-First have nearly identical mutual information (1.1344 vs. 1.1358 nats) but very different effective model results. The SSM excels on Copy-Last (0.8125 BC) because the last token's information is immediately available to the recurrent state, while Copy-First requires retaining the first token through all subsequent steps.

4 DISCUSSION

4.1 Viability of Non-Transformer Effective Models

Our results demonstrate that non-transformer architectures can serve as viable effective models for a significant class of LLM behaviors. The SSM achieves 0.8125 behavioral consistency on Copy-Last with only 928 parameters—less than one-third the 3200 parameters required by the Transformer baseline, which itself only reaches 0.4844 BC. This establishes that for tasks with favorable memory-access patterns, SSMs are *more parameter-efficient* effective models than transformers.

4.2 Perturbation Framework Extension

The SSM perturbation analysis reveals a more distributed structure than the transformer case assumed by Raju et al. [9]. Rather than concentrating in a low-rank "ideal" component with small perturbation, the SSM's projections utilize most of their available dimensions (effective dimension 11.1061–12.3633 out of 16). This suggests that SSM-based effective models may require a different perturbation theory—one based on spectral properties of the state transition matrix A rather than a simple additive decomposition.

The spectral radius provides a natural complexity measure for SSM effective models: simpler tasks (Copy-Last, $\rho = 0.192$) require less recurrent memory than complex tasks (Reverse-Sum, $\rho = 0.3529$).

4.3 Task Complexity Determines Architecture Choice

The mutual information $I(X;Y)$ provides a useful predictor of effective modeling difficulty. Tasks with low mutual information (Pattern-Detect, 0.5805 nats) are well-served by all architectures, including the fixed-receptive-field TCN. Tasks with high mutual information requiring compositional reasoning (Reverse-Sum, 1.1348 nats) challenge all small effective models regardless of architecture.

The key finding is that the *nature* of the task—not just its information-theoretic complexity—determines which architecture succeeds. Copy-Last and Reverse-Sum have similar mutual information but very different architecture rankings, because they differ in the computational primitives required (memory access vs. arithmetic composition).

4.4 Limitations and Future Work

Our experiments use randomly initialized models rather than trained/distilled models, testing whether the architecture's inductive bias alone provides agreement with the teacher. Training-based distillation would likely improve all results and may change the relative rankings. The vocabulary and sequence length are small ($V = 4$, $T = 3$); scaling to realistic LLM settings is an important direction. Finally, our teacher is synthetic; future work should distill from actual LLMs.

5 CONCLUSION

We investigate whether non-transformer architectures can serve as effective models for capturing LLM behavior under fixed prompts and tasks. Through a systematic multi-architecture distillation competition across five tasks, we find that SSMs achieve the highest behavioral consistency (0.8125) on memory-access tasks with the fewest parameters (928), while all architectures struggle with compositional tasks (best BC 0.3125 on Reverse-Sum). SSM parameters admit a natural perturbation decomposition, though with a more distributed structure (Frobenius ratios 0.8797–0.9162) than the low-rank ideal assumed by transformer effective model theory. Task mutual information ($I(X;Y)$ ranging from 0.5805 to 1.1358 nats) predicts effective modeling difficulty, but task structure determines which architecture succeeds. These results establish that the effective model framework of Raju et al. extends beyond transformers, with SSMs as the most promising non-transformer alternative for a significant class of LLM behaviors.

REFERENCES

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271* (2018).
[2] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* (2014).
[3] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752* (2023).
[4] Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*.
[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).
[6] Yoon Kim and Alexander M Rush. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1317–1327.
[7] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting Recurrent Neural Networks for Long Sequences. *arXiv preprint arXiv:2303.06349* (2023).
[8] Bo Peng, Eric Alcaide, Quentin Anthony, et al. 2023. RWKV: Reinventing RNNs for the Transformer Era. *arXiv preprint arXiv:2305.13048* (2023).
[9] Rajesh Raju et al. 2026. A model of errors in transformers. *arXiv preprint arXiv:2601.14175* (2026).