Cross-Architecture Centered Kernel Alignment Reveals Where Sequence Models Converge and Diverge on Synthetic Algorithmic Tasks

Harsh Rathva [1]

Abstract

Mechanistic interpretability has advanced primarily through architecture-specific case studies, particularly in Transformers, while recurrent models are typically compared only at the behavioral or formal-theoretic level. A basic empirical question therefore remains open: when matched-capacity sequence models are trained on the same controlled task, do they converge to aligned internal representations? We address this question with a compact benchmark comparing two-layer Transformers, LSTMs, and GRUs on five synthetic tasks spanning modular arithmetic, formal languages, and associative retrieval. For each architecture pair, we compute layer-wise centered kernel alignment (CKA) heatmaps averaged over eight random seeds, together with within-architecture seed baselines, a normalized convergence index, and layer-wise linear probes. Three findings emerge. First, recurrent models (LSTM, GRU) substantially outperform the Transformer under our fixed training protocol, creating an asymmetric-competence regime that must be accounted for when interpreting similarity scores. Second, under this fixed protocol, recurrent–recurrent alignment consistently exceeds Transformer–recurrent alignment, particularly on hierarchical tasks (Dyck-2), where the convergence index drops to 0.55–0.68 across families versus 0.90 within the recurrent family; however, a pilot grokking experiment shows that much of this gap narrows when the Transformer is trained to competence, suggesting that competence differences are a major driver of the observed similarity gap. Third, moderate cross-architecture CKA persists even when all models perform near chance, indicating that geometric similarity can reflect shared input structure or dataset geometry (Cui et al., 2022) rather than shared mechanistic solutions. These results suggest that representational convergence is task-contingent rather than universal and that CKA is most informative when interpreted jointly with task performance, probing accuracy, and within-architecture baselines, and future work should incorporate calibrated similarity measures (Murphy et al., 2024; Gröger et al., 2026) to disentangle signal from null-floor effects.

[1]Sardar Vallabhbhai National Institute of Technology (SVNIT), Surat, India. Correspondence to: Harsh Rathva <u24ai036@aid.svnit.ac.in>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Mechanistic interpretability aims to reverse-engineer the algorithms implemented by neural networks from their internal activations and weights. The most detailed results to date concern Transformer-specific circuits, including induction heads for in-context learning (Olsson et al., 2022) and Fourier-based circuits for modular arithmetic (Nanda et al., 2023). Recurrent architectures, by contrast, are more commonly studied through their inductive biases or formal expressive limits (Hochreiter & Schmidhuber, 1997; Cho et al., 2014; Wen et al., 2024) than through direct comparison of their learned representations with those of attention-based models.

This asymmetry raises a practical question for the field. If different architectures converge to similar internal geometries on a shared task, then mechanistic explanations discovered in one family may transfer to another. If they do not, then architecture-specific tools remain necessary even for small, controlled problems. Behavioral accuracy alone cannot resolve the question: two models may achieve identical accuracy through different internal representations, or may exhibit similar geometry while relying on shallow heuristics.

We study this question in a deliberately constrained setting. We train matched two-layer Transformer, LSTM, and GRU classifiers on five synthetic tasks: modular addition, Dyck-1, Dyck-2, induction, and associative recall, using a shared codebase, identical hyperparameters, and eight random seeds. We then compute layer-wise CKA

heatmaps (Kornblith et al., 2019) between all architecture pairs, compare them against within-architecture seed baselines, and supplement the geometric analysis with layer-wise linear probes.

Our contribution is an empirical benchmark for cross-architecture representational comparison. The benchmark is compact enough for direct inspection yet broad enough to expose qualitatively distinct regimes. We report three principal findings, detailed in section 4: an asymmetric-competence regime that complicates CKA interpretation, stronger within-family than cross-family alignment, and moderate CKA persisting even at chance-level performance.

2. Related Work

Mechanistic interpretability in Transformers. Olsson et al. (2022) identify induction heads as a core mechanism for in-context learning in Transformers, while Nanda et al. (2023) reverse-engineer modular-addition circuits and connect them to Fourier structure. These studies motivate our task selection and provide architecture-specific hypotheses against which cross-architecture alignment can be evaluated.

Representation similarity. CKA (Kornblith et al., 2019) is a standard metric for comparing learned representations because of its invariance properties. However, Cui et al. (2022) show that CKA can be confounded by shared dataset geometry, particularly in shallow layers, and Murphy et al. (2024) demonstrate that raw linear CKA carries a non-trivial positive bias at finite dimensions. Gröger et al. (2026) further show that spectral (global) and neighborhood-based (local) similarity metrics can yield qualitatively different conclusions on the same activations. We therefore interpret CKA comparatively, relative to within-architecture baselines and downstream accuracy, rather than treating any single score as mechanistically definitive.

Recurrent architectures and their limits. LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014) improve long-range memory in recurrent models. Wen et al. (2024) establish that in-context retrieval is a fundamental bottleneck for recurrent architectures relative to Transformers. Suzgun et al. (2019) show that recurrent memory is well-suited to hierarchical structure in Dyck languages. Our work complements these behavioral and theoretical analyses with layer-wise representational comparison.

Representation convergence across architectures. Huh et al. (2024) argue that diverse models converge to shared representations as scale and data increase (the "platonic representation hypothesis"). Bansal et al. (2021) use model stitching to test functional compatibility of representations across trained models. Our work tests a related question at the small-model, controlled-task level relevant to mechanistic interpretability.

3. Experimental Setup

3.1. Tasks

We evaluate five synthetic sequence-classification tasks:

- **Modular addition:** predict $(a + b) \bmod 10$ from tokenized equations (10 classes).
- **Dyck-1:** classify single-bracket sequences as balanced or corrupted (binary).
- **Dyck-2:** the same setting with two bracket types (binary).
- **Induction:** retrieve the value associated with a queried key from a short key–value prompt.
- **Associative recall:** retrieve a value from a longer key–value list separated by delimiters.

Each task uses 2,048 training and 512 validation examples with a fixed dataset seed.

3.2. Models and Training

All architectures use two layers and hidden dimension 128. The Transformer uses four attention heads with sinusoidal positional encodings. LSTM and GRU models use stacked recurrent blocks with the same hidden size.[1] All models share the same optimizer (AdamW), learning rate (10^{-3}), weight decay (10^{-4}), batch size (64), and training duration (10 epochs). We train eight random seeds (42–49) per architecture per task. We acknowledge that this fixed-hyperparameter protocol disadvantages the Transformer relative to recurrent models (section 6). Additional grokking and activation-patching analyses are reported as follow-up pilot studies (sections 4.4 and 4.6) rather than fully integrated 8-seed benchmark results.

3.3. Activation Extraction and CKA

We extract activations at three checkpoints (embedding output, layer 1, layer 2) on the validation set, mean-pooled across sequence positions to produce a single vector per example (i.e., $\bar{h} = \frac{1}{T}\sum_{t=1}^{T} h_t$). This pooling strategy ensures a consistent representation format across architectures, since Transformers lack a single "final" hidden state. Linear CKA

[1] Approximate encoder parameter counts (excluding shared embedding and classifier head): Transformer ≈397K, LSTM ≈264K, GRU ≈198K. The Transformer is largest due to its multi-head attention and feed-forward sub-layers.

between activation matrices $X \in \mathbb{R}^{n \times d_x}$ and $Y \in \mathbb{R}^{n \times d_y}$ is

$$\mathrm{CKA}(X, Y) = \frac{\langle \widetilde{K}_X, \widetilde{K}_Y \rangle_F}{\|\widetilde{K}_X\|_F \, \|\widetilde{K}_Y\|_F}, \tag{1}$$

where $\widetilde{K}_X, \widetilde{K}_Y$ are centered Gram matrices (Kornblith et al., 2019). Cross-architecture heatmaps are averaged over all seed pairs; within-architecture baselines are computed from the $\binom{8}{2} = 28$ seed pairs per architecture.

3.4. Convergence Index

To normalize cross-architecture CKA against within-architecture variability, we define the *convergence index* (CI) as the ratio of mean cross-architecture CKA to the geometric mean of the two corresponding within-architecture baselines:

$$\mathrm{CI}(A, B) = \frac{\overline{\mathrm{CKA}}_{A \leftrightarrow B}}{\sqrt{\overline{\mathrm{CKA}}_{A \leftrightarrow A} \cdot \overline{\mathrm{CKA}}_{B \leftrightarrow B}}}. \tag{2}$$

The geometric mean in the denominator is chosen because it treats both within-architecture baselines symmetrically and reduces to the single baseline when $A = B$. It also corresponds to the normalization used in the Pearson correlation of Gram matrices, ensuring that $\mathrm{CI} = 1$ when cross-family alignment equals within-family consistency.[2] A value near 1 indicates that cross-family alignment is comparable to within-family consistency; lower values indicate architecture-specific representations.

3.5. Linear Probing

At each layer, we fit a logistic-regression probe (L-BFGS, ℓ_2 regularization $C = 1.0$) on cached activations to predict task-relevant targets: the answer class for modular addition, maximum nesting depth for Dyck tasks, and the target value for induction and associative recall. Probe accuracy complements CKA by testing whether the geometric structure captured by CKA corresponds to linearly accessible task-relevant information.

4. Results

4.1. Accuracy: An Asymmetric-Competence Regime

Table 1 reveals a pronounced architecture effect. Recurrent models solve modular addition and Dyck-1 near-perfectly and achieve strong performance on Dyck-2, while the Transformer achieves only chance-level accuracy on modular addition (0.100 ± 0.014 for a 10-class task) and substantially lower accuracy on all remaining tasks. Zero standard deviation on Dyck-1 across eight seeds suggests the Transformer converges to a fixed degenerate predictor.

Task	Chance	Transformer	LSTM	GRU
Mod. Add.	0.10	0.100 ± 0.014	0.990 ± 0.016	0.995 ± 0.012
Dyck-1	0.50	0.768 ± 0.000	1.000 ± 0.000	0.859 ± 0.108
Dyck-2	0.50	0.793 ± 0.000	0.986 ± 0.009	0.888 ± 0.074
Induction	0.25	0.252 ± 0.007	0.529 ± 0.017	0.599 ± 0.069
Assoc. Recall	0.25	0.237 ± 0.028	0.352 ± 0.016	0.328 ± 0.033

Table 1. Validation accuracy (mean $\pm$ std over 8 seeds). The Transformer underperforms recurrent models on all tasks under the shared training protocol, including chance-level performance on modular addition.

This performance gap likely reflects our deliberately fixed training protocol: Transformers on small algorithmic tasks typically require longer training, warmup schedules, or curriculum strategies (Nanda et al., 2023). We retain this regime intentionally because the resulting asymmetric-competence setting exposes an important interpretive pitfall: cross-architecture CKA between a competent and an incompetent model measures geometric overlap between a learned solution and a degenerate one, which should not be confused with evidence of convergent algorithms.

4.2. CKA: Geometric Similarity Does Not Imply Competence

Figure 1 presents cross-architecture CKA heatmaps. Despite the Transformer's near-chance accuracy on modular addition, Transformer–LSTM CKA at the embedding layer remains 0.70 and Transformer–GRU CKA reaches 0.75. This moderate alignment reflects shared input structure (identical tokenized inputs processed through randomly initialized but similarly-structured embedding layers) rather than shared algorithmic solutions. At deeper layers, Transformer–recurrent CKA decays to 0.20–0.24, consistent with divergent representations downstream of a non-functional Transformer.

By contrast, LSTM–GRU CKA on modular addition remains 0.23–0.71 across layers, and both models achieve $>99\%$ accuracy. The recurrent–recurrent comparison thus captures alignment between two competent solutions, while Transformer–recurrent comparisons capture alignment between a competent solution and a degenerate one.

4.3. Convergence Index: Recurrent Alignment Dominates

Table 2 quantifies the cross-vs-within alignment gap. LSTM–GRU convergence indices are near or above 1.0 on all tasks, indicating that cross-family alignment within the recurrent family matches within-architecture seed consistency. Transformer–recurrent CI is lower across all tasks and drops most sharply on Dyck-2 (0.550 for T↔LSTM) and associative recall (0.572), the two tasks with the largest accuracy gaps.

[2] The arithmetic mean $\frac{1}{2}(\overline{\mathrm{CKA}}_{A \leftrightarrow A} + \overline{\mathrm{CKA}}_{B \leftrightarrow B})$ is an alternative; in practice it yields nearly identical rankings across our task–pair combinations.

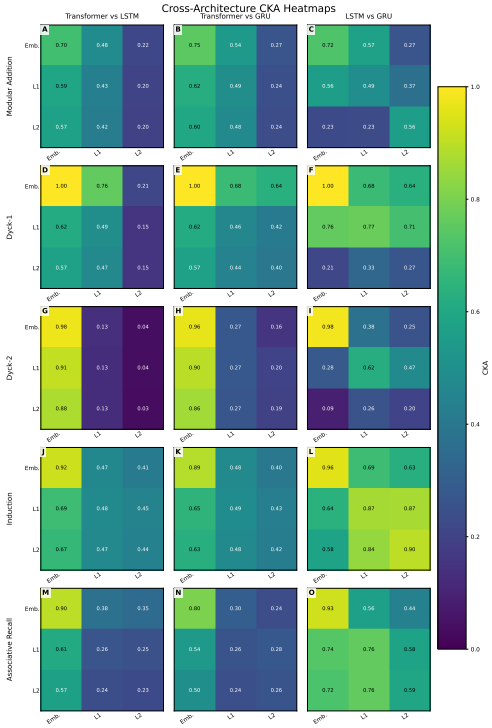

Figure 1. Cross-architecture CKA heatmaps (mean over 8 seeds). Rows: tasks; columns: architecture pairs. Embedding-layer CKA is high across all pairs and tasks, reflecting shared input structure. Deeper-layer CKA decays fastest for Transformer–recurrent pairs, particularly on Dyck-2 and associative recall. LSTM–GRU alignment remains consistently higher than Transformer–recurrent alignment.

Task	T↔L	T↔G	L↔G
Mod. Add.	0.795	0.882	1.377
Dyck-1	0.821	0.887	0.932
Dyck-2	0.550	0.680	0.901
Induction	0.765	0.792	0.988
Assoc. Recall	0.572	0.626	0.955

Table 2. Convergence index (CI) by task and architecture pair. LSTM↔GRU CI is consistently highest, indicating stronger within-family than cross-family alignment. Values below 1.0 indicate weaker alignment than the within-architecture baseline.

The Dyck-2 pattern is notable because it persists even though the Transformer achieves moderate accuracy (0.793). This suggests that hierarchical bracket tracking induces recurrent-specific geometry even when the Transformer partially succeeds, consistent with the hypothesis that recurrent state updates are a natural fit for counter-like or stack-like structure (Suzgun et al., 2019).

The LSTM–GRU CI exceeding 1.0 on modular addition (1.377) indicates that these architectures converge to more similar representations *across families* than individual architectures produce *across seeds*, consistent with a tightly constrained solution manifold that is more sensitive to random initialization than to the LSTM-vs-GRU architectural difference.

Within-architecture seed baselines (28 pairs per architecture; section B) confirm that same-family consistency is generally stronger than cross-family alignment, particularly on structured tasks.

4.4. Pilot Competent-Transformer Comparison

The low Transformer–recurrent alignment observed under the default protocol could reflect either genuine representational incompatibility or the Transformer's failure to learn. To probe this ambiguity, we ran a follow-up *pilot* grokking-style experiment on modular addition using 500 epochs with weight decay 1.0. Because the full 8-seed rerun was not complete at manuscript freeze, we report this follow-up as a single-run comparison rather than as a seed-aggregated benchmark result.

The pilot result is informative nonetheless. Transformer accuracy rises to 0.942, and the Transformer–LSTM layer-2 CKA increases from 0.20 under the short protocol to 0.68 under the grokking-style protocol (table 3). This pattern supports the interpretation that competence is a major driver of the cross-architecture similarity gap. At the same time, we avoid a stronger claim because the grokked comparison is presently a follow-up pilot rather than an 8-seed extension of the main benchmark.[3]

[3]Code: github.com/ezylopx5/Cross-Architecture-CKA-on-Synthetic-Algorithmic-Tasks

Protocol	Seeds	T Acc.	T↔L	T↔G
Short (10 ep.)	8	0.100 ± 0.014	0.20	0.24
Grokked (500 ep.)	1	0.942	0.68	—

Table 3. Pilot modular-addition follow-up comparing the default benchmark protocol with an extended grokking-style Transformer run. The short protocol is the main 8-seed benchmark; the grokked row is a single-run pilot. The missing grokked T↔G cell is deferred to the full 8-seed extension.

4.5. Probing: Functional Information Tracks Competence

Figure 2 shows layer-wise linear-probe accuracy. On modular addition, recurrent models reach near-perfect probe accuracy by layer 1, while the Transformer rises only to 0.89 at layer 2. This partial probe success despite chance-level classification accuracy indicates that the Transformer's representations encode task-relevant structure that the classification head fails to exploit, a dissociation between representational content and readout competence.

On Dyck tasks, all architectures show above-chance probing for nesting depth, but recurrent models reach ceiling earlier. The majority-class baseline for Dyck-2 is 0.50, indicating that the Transformer's 0.793 accuracy reflects meaningful learning. On induction, GRU and LSTM probe accuracy reaches 0.85 at deeper layers while Transformer probes plateau at 0.56, mirroring the task-accuracy gap.

4.6. Pilot Activation Patching: Preliminary Functional Overlap

To test whether high LSTM–GRU CKA on Dyck-2 reflects functional equivalence rather than superficial geometry, we perform a cross-architecture interchange intervention. We fit a linear projection from GRU layer-1 activations to LSTM layer-1 activations on half the validation set, then replace the LSTM's layer-1 representations with projected GRU activations on the held-out half.

The patched model retains 0.862 accuracy versus 0.988 for the unpatched LSTM, a 12.8% relative drop, which we interpret as evidence of *partial* functional overlap. The drop could reflect either genuine representational misalignment in the non-shared subspace or limitations of the linear projection itself: a 128-to-128 linear map is expressive enough to absorb substantial structure, so the residual drop may understate the true representational gap. Conversely, the high retained accuracy suggests that the dominant computational motifs (bracket counting, depth tracking) are encoded in a shared linear subspace across LSTM and GRU, consistent with their high CKA.

This result is currently a single-pair pilot; seed-averaged patching with random and low-CKA controls (e.g., Transformer→LSTM patching, which would serve as a negative control given the lower cross-family CKA) remains for future work.

Source → LSTM	CKA	Unpatched	Patched	Δ
GRU (high CKA)	0.70	0.988	0.862	−12.8%

Table 4. Pilot activation-patching result on Dyck-2 at layer 1. The row reports a single held-out split for one trained LSTM–GRU pair.

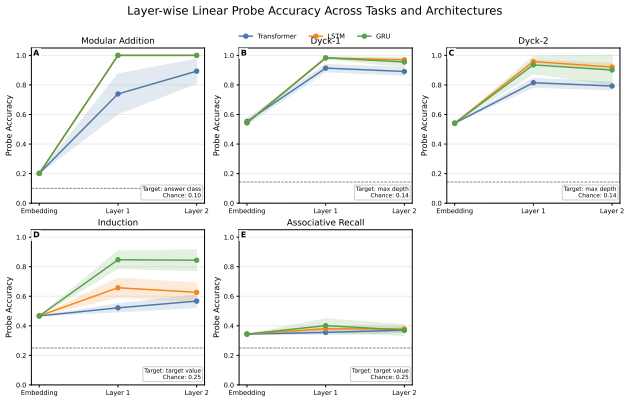

Figure 2. Layer-wise linear-probe accuracy. Shaded bands show ±1 std over 8 seeds. Dashed lines indicate chance level. Recurrent models achieve higher probe accuracy at deeper layers, consistent with their superior task accuracy. The Transformer shows partial probe success on modular addition despite chance-level classification, suggesting representational content that the classification head fails to decode.

5. Discussion

Our results point to a simple but consequential framing: *representational convergence across sequence architectures is task-contingent and regime-dependent.*

The asymmetric-competence regime produced by our fixed training protocol exposes an interpretive hazard: moderate cross-architecture CKA can persist even when one model has not learned the task, driven by shared input geometry rather than shared algorithms. This caution is complementary to that of Cui et al. (2022), who show that CKA can reflect dataset structure, and to Murphy et al. (2024), who demonstrate that raw linear CKA carries a positive bias at finite dimensions that can inflate values near the null floor. In our setting, embedding-layer CKA is consistently high regardless of downstream competence, suggesting that it captures tokenization-level rather than computation-level similarity. Future work should apply permutation-calibrated CKA (Gröger et al., 2026) or debiased estimators (Murphy et al., 2024) to separate signal from null-floor inflation, particularly at deeper layers where cross-family values approach the bias floor.

When both models are competent, as in recurrent–recurrent comparisons on modular addition and Dyck tasks, CKA

and probing together provide a more informative picture. The high LSTM–GRU CI (>0.9 on most tasks) suggests that these architectures converge to closely aligned internal representations, consistent with their shared inductive bias for sequential state maintenance.

Dual-regime perspective. Our benchmark naturally decomposes into two complementary regimes: a *same-protocol* comparison (all architectures trained identically, exposing competence asymmetries) and a *matched-competence* comparison (the grokking pilot in section 4.4, where the Transformer is trained to convergence). Neither regime alone resolves the architecture-vs-competence question. The same-protocol regime shows that CKA gaps can be large when competence differs; the grokking pilot shows that much of this gap closes when competence is equalized (T↔LSTM CKA rises from 0.20 to 0.68). Future work should treat both regimes as first-class, since the *contrast* between them is itself informative: it quantifies how much of the representational gap is attributable to learning dynamics versus architectural inductive bias.

Two practical implications follow for mechanistic interpretability. First, under matched training, mechanistic explanations are more likely to transfer within architectural families (e.g., LSTM to GRU) than across families (e.g., Transformer to LSTM), though the degree of this gap may narrow with architecture-specific tuning. Second, CKA is most valuable as a triage tool, identifying where closer circuit-level analysis is likely to transfer, rather than as standalone evidence of mechanistic alignment. Complementing CKA with local similarity metrics (e.g., mutual k-NN) and causal interventions would provide a more complete picture of when geometric alignment reflects functional equivalence (Gröger et al., 2026).

6. Limitations and Future Work

Transformer underperformance. The most significant limitation is the Transformer's failure to learn several tasks under our fixed training protocol. Transformers on modular addition are known to require extended training and warmup schedules for grokking (Nanda et al., 2023). Our protocol (10 epochs, no warmup, shared learning rate) is likely insufficient. Future iterations should include architecture-specific tuning or extended training to ensure that cross-architecture comparisons involve models of comparable competence. In the present work, we report the asymmetric regime as-is and interpret it explicitly.

Scale and scope. We use only two-layer models with hidden dimension 128 on short synthetic tasks. Whether the patterns we observe, particularly the strong recurrent–recurrent alignment, generalize to larger models, longer sequences, or natural-language data remains open.

Methodological limitations. Mean-pooled activations and linear CKA may obscure token-level or circuit-level structure. We use mean-pooled activations for consistency across architectures (Transformers lack a single "final" hidden state). For recurrent models, final-timestep activations may better capture task-relevant state; we leave this comparison to future work. CKA can be confounded by shared dataset geometry (Cui et al., 2022). Future work should pair CKA with causal interventions (activation patching, ablation) to test whether aligned geometry corresponds to aligned circuits.

Follow-up analyses. The grokking and activation-patching studies are currently pilot follow-ups rather than full 8-seed benchmark extensions. They are useful for stress-testing the main interpretation, but their numerical results should be read as preliminary until the full reruns and control patches are complete.

Missing architectures. Modern state-space models (e.g., Mamba) occupy an intermediate position between Transformers and RNNs. Including them would clarify whether the recurrent-family alignment patterns we observe extend to this emerging class.

7. Conclusion

We presented a compact benchmark for comparing Transformer, LSTM, and GRU representations across five synthetic algorithmic tasks using CKA heatmaps, linear probes, and convergence indices. Under a fixed training protocol, representational convergence is selective: recurrent models align strongly with one another across tasks, while Transformer–recurrent alignment is weaker and task-dependent. However, a pilot grokking experiment demonstrates that much of the Transformer–recurrent gap narrows when the Transformer is trained to competence, indicating that competence differences are a major driver of the observed similarity gap. Moderate geometric similarity can also coexist with large competence gaps, underscoring the need to interpret CKA jointly with task performance and calibrated baselines (Murphy et al., 2024). We hope this benchmark serves both as a reproducible artifact and as a reminder that mechanistic explanations should be validated across architectures, not assumed to transfer.

Impact Statement

This work advances mechanistic interpretability through controlled representational comparison. Because it is limited to small synthetic tasks, it does not directly enable deployment or misuse, but may contribute to methods for

analyzing larger models.

References

Bansal, Y., Nakkiran, P., and Barak, B. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems 34*, 2021.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Cui, T., Kumar, Y., Marttinen, P., and Kaski, S. Deconfounded representation similarity for comparison of neural networks. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 19138–19151. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/79cbf4f96c2bcc67267421154da689dd-Paper-Conference.pdf.

Gröger, F., Wen, S., and Brbić, M. Revisiting the platonic representation hypothesis: An aristotelian view. In *International Conference on Machine Learning (ICML)*, 2026.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Huh, M., Cheung, B., Wang, T., and Isola, P. Position: The platonic representation hypothesis. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20617–20642. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huh24a.html.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3519–3529, 2019.

Murphy, A., Zylberberg, J., and Fyshe, A. Correcting biased centered kernel alignment measures in biological and artificial neural networks. *arXiv preprint arXiv:2405.01012*, 2024.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Suzgun, M., Gehrmann, S., Belinkov, Y., and Shieber, S. M. Memory-augmented recurrent neural networks can learn generalized dyck languages. *arXiv preprint arXiv:1911.03329*, 2019.

Wen, K., Dang, X., and Lyu, K. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. *arXiv preprint arXiv:2402.18510*, 2024.

A. Implementation Details

All experiments use a shared training pipeline. The Transformer uses learned token embeddings, sinusoidal positional encodings, and two encoder layers with GELU activations. LSTM and GRU models use stacked single-layer recurrent blocks. Activations are mean-pooled over sequence positions before CKA computation. Training uses an MLX backend on Apple Silicon; probes are fit in PyTorch on cached activations. Full configuration files and metric tables are included in the supplementary material.

B. Within-Architecture Seed Baselines

Figure 3 shows within-architecture CKA computed over all 28 seed pairs per architecture. These baselines provide the denominator for the convergence index (table 2) and serve as the reference against which cross-architecture alignment should be evaluated.

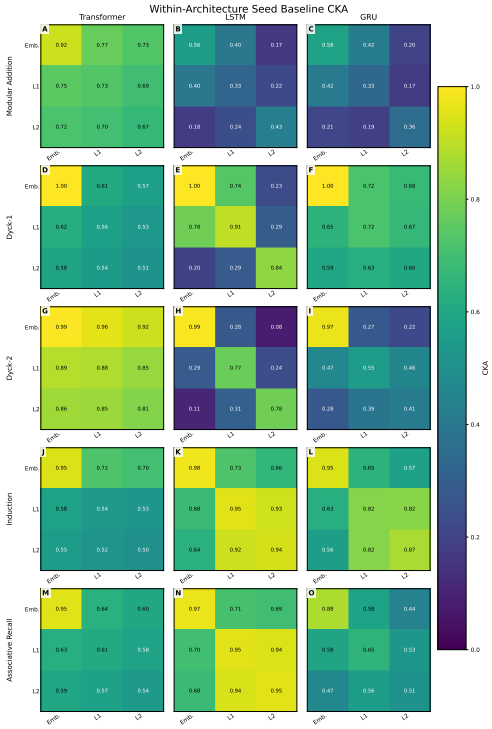

Figure 3. Within-architecture CKA baselines (28 seed pairs per architecture). Same-architecture consistency is highest for the Transformer on modular addition (0.72–0.92) despite its chance-level accuracy, suggesting that degenerate solutions can be highly reproducible. LSTM and GRU show moderate within-architecture consistency, particularly at deeper layers on harder tasks.

C. Accuracy Summary

Figure 4 reproduces the accuracy landscape from table 1 in visual form, with individual seed results shown as scatter points. Chance levels are indicated by dashed lines for each task.

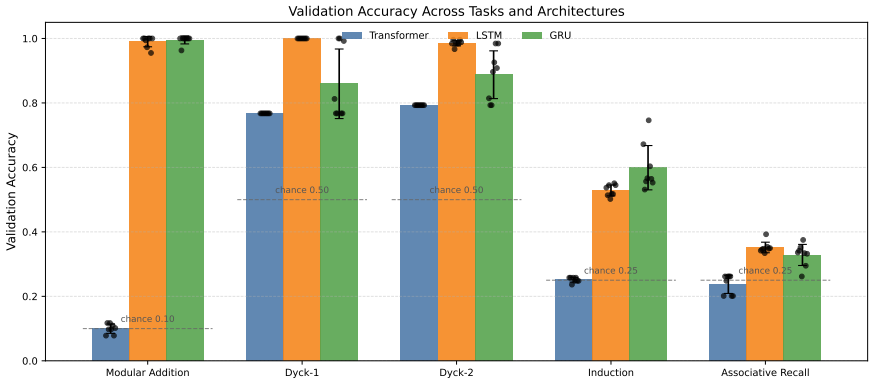

Figure 4. Validation accuracy across tasks and architectures (8 seeds). Error bars show ± 1 std; dots show individual seeds. Recurrent models dominate all tasks under the shared training protocol.