Higher Embedding Dimension Creates a Stronger World Model for a Simple Sorting Task

Brady Bhalla
California Institute of Technology
Pasadena, CA 91125
bbhalla@caltech.edu

Honglu Fan
Google DeepMind
London, GB
honglu.fan@unige.ch

Nancy Chen
Cornell University
Ithaca, NY 14583
nc494@cornell.edu

Tony Yue YU
California Institute of Technology
Pasadena, CA 91125
yuyuetony@gmail.com

Abstract

We study how embedding dimension affects the emergence of an internal "world model" in a transformer trained with reinforcement learning to perform bubble-sort-style adjacent swaps. While even very small embedding dimensions are sufficient for models to achieve high accuracy, larger dimensions yield representations that are more faithful, consistent, and robust. In particular, higher embedding dimensions strengthen the formation of structured internal representation and leads to better interpretability. After hundreds of experiments, we observe two consistent mechanisms: (1) the last row of the attention weight matrix monotonically encodes the global ordering of tokens; and (2) the selected transposition aligns with the largest adjacent difference of these encoded values. Our results provide quantitative evidence that transformers build structured internal world models and that model size improves representation quality in addition to end performance. We release metrics and analyses that can be reused to probe similar tasks.

1 Introduction

Sorting has long been a canonical problem in computer science, valued both for its theoretical significance and its ubiquity in practice [5]. There are numerous trade-offs between different sorting algorithms, and sorting as efficiently as possible is extremely important because of how often the algorithm is needed [7]. Beyond efficiency concerns, sorting also serves as a natural testbed for studying how algorithms—and, increasingly, learned models—structure computation. Recent advances have shown that deep learning can yield new insights into traditional computer science problems such as sorting, even leading to improvements that have been incorporated into LLVM standard sort for C++ [15]. Our interest, however, does not fall into algorithmic acceleration, but rather the *mechanisms* by which small neural architectures internally represent and execute discrete procedures.

Transformers are now the dominant architecture for sequence modeling [22], achieving remarkable capabilities in language, reasoning, and beyond [13]. Logical and mathematical benchmarks highlight their strengths but also reveal persistent reasoning limitations [4]. While strategies such as chain-of-thought prompting mitigate some failures [25, 23], these remain workarounds rather than evidence of robust reasoning.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability Workshop at NeurIPS 2025.

In parallel, *mechanistic interpretability* has emerged as a program for understanding the inner workings of neural networks [3]. Small transformer networks without multilayer perceptrons in particular have been found to be fairly amenable to analysis [6]. Related work on grokking shows that transformers can form generalizable latent representations of tasks [14]. This behavior connects to the broader "world model" hypothesis: neural networks build structured internal representations of their environment [9]. Encouraging the development of a world model has has been shown to improve reasoning abilities in large language models (LLMs) [10].

In this paper, we investigate whether small transformer models trained via reinforcement learning (RL) develop an interpretable world models of the bubble sort process, and how it differs with respect to varying embedding dimensions. Instead of phrasing it as an autoregressive generation problem, we recast it in a reinforcement learning framework and take advantage of the interpretability of the transformer to gain insight into how the model solves the task. Our reinforcement learning setting has states defined by permutations of tokens representing numbers from 1 to n. The allowed actions were limited to swaps between adjacent elements in the input sequence, with a reward given once the sequence became fully sorted. This was trained using the Proximal Policy Optimization (PPO) algorithm [21]. Our contributions are as follows:

- We introduce sorting as a reinforcement-learning testbed for mechanistic interpretability.
- We provide an empirical study showing that increasing embedding dimension substantially improves representation faithfulness, consistency, and robustness.
- We identify two consistent patterns—a global order encoding in attention weights and largest difference selection rule—that explain agent behavior.
- We release metrics and methodology for evaluating alignment between hypothesized mechanisms and model internals.

2 Related works

2.1 Transformer-based world models in reinforcement learning

Recent research has explored the use of transformer architectures to build world models that enhance data efficiency in reinforcement learning. For example, IRIS combines a discrete autoencoder with an autoregressive transformer to learn efficient world models capable of achieving human-level performance on Atari within only two hours of gameplay [17]. Similarly, STORM integrates stochastic transformers with variational components to enhance model-based RL [26].

2.2 Mechanistic interpretability in neural systems and toy environments

Mechanistic interpretability refers to identifying internal, algorithmically meaningful structures within models. Landmark studies in transformer interpretability, such as those by the Anthropic Clarity team, have revealed algorithmic behaviors, such as induction heads, in small-scale transformers [20]. More recent work, including analysis of Othello-GPT, demonstrates that linear latent world representations can form even without explicit supervision [19]. In parallel, studies such as SortBench evaluate how well large language models can capture and execute simple algorithmic behaviors like sorting [11].

2.3 Representation capacity and embedding dimension

The question of how model capacity—especially embedding size—affects representation quality has been widely studied. For example, Word2vec models exhibit improved embedding quality with increased dimensionality, up to a point of diminishing returns [18]. Empirical studies show that higher-dimensional embeddings often enhance intrinsic task performance, though extrinsic tasks may require careful tuning [16]. Theoretical work further reveals a bias–variance trade-off that shapes optimal embedding dimensionality [24].

3 Preliminaries and discussions

3.1 Transformers

Transformers [22] have become the dominant architecture for sequence modeling, excelling in language, vision, and reinforcement learning. Their core mechanism, self-attention, computes interactions between all pairs of tokens in a sequence, enabling flexible representation of order and relational structure. While modern applications often rely on deep, multi-head architectures, even single-head, shallow transformers are capable of learning structured, interpretable behaviors. This makes them ideal candidates for mechanistic interpretability studies in constrained settings. In this work, we deliberately use minimal transformer architectures to isolate the role of embedding dimension in shaping internal representations (details of the model architecture are provided in Appendix 8.1). By reducing architectural complexity, we can more directly attribute observed world-model–like behavior to the interaction of self-attention and reinforcement learning dynamics.

3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [21] is a reinforcement learning algorithm widely adopted for its balance between stability and performance. PPO constrains policy updates using a clipped surrogate objective, which prevents destructive shifts in behavior while still encouraging steady improvement. Its simplicity and robustness have made it a standard choice across domains ranging from Atari to robotics, and more recently as the backbone of reinforcement learning with human feedback in large language models. In our setting, PPO provides a practical way to train agents to discover sorting strategies without prescribing explicit rules. Because PPO naturally encourages agents to form structured latent representations of their environment, while maintaining training stability, it is particularly well-suited for probing whether interpretable world models emerge in a toy sorting task.

3.3 Notation and model setup

The input to the model consists of tokens corresponding to numerical values from 1 to ℓ, ordered by some permutation π. The transformer uses single-headed self-attention, with queries Q, keys K of dimension d_k, and values V. With this model, the attention weights are given by

$$W = \frac{QK^\top}{\sqrt{d_k}}.$$

3.4 Why sorting as a testbed

Our work builds upon a growing body of research that uses toy problems as controlled settings for mechanistic interpretability studies. By constraining the complexity of the task and environment, researchers can systematically probe model internals without the confounding variability present in real-world data [6, 3]. Sorting, despite its apparent simplicity, is a particularly appealing choice because its state space is finite, its optimal solutions are well-defined, and its intermediate states have a clear, human-interpretable structure. This makes it possible to directly compare the agent's learned representations to ground truth abstractions such as "global order" or "adjacent differences." Unlike real-world RL tasks, there is no ambiguity in goal definition or reward signals, which allows us to attribute observed behaviors more confidently to architectural properties rather than to environmental noise.

Unlike prior work that uses deep RL to search for efficient new algorithms, our aim is not to improve performance but to understand the mechanisms a network employs to solve a task it could already solve [15]. We deliberately choose to use a reinforcement learning setup because RL-trained models tend to develop internal state representations better aligned with an environment's causal structure, resonating with the "world model" hypothesis in deep learning [9]. In our case, the emergent ordering circuit in the attention weights can be seen as a minimalist form of such a model.

Furthermore, there is precedent for embedding dimension playing a critical role in representation quality across modalities. In natural language processing, larger hidden sizes often permit more nuanced syntactic and semantic encoding [12]. By exploring this axis in a highly constrained RL setting, we aim to isolate how embedding dimension affects the fidelity of an interpretable, algorithmically relevant latent structure.

3.5 Empirical observations

Exploratory experiments revealed two consistent behaviors in trained agents:

1. **A global order encoding in attention weights:** the final row of the attention matrix appears to monotonically track the input tokens' global order (Observation 1).
2. **A difference-based rule for swap selection:** the chosen transposition tends to correspond to the largest adjacent difference between these encoded values (Observation 2).

4 Methods

4.1 Model evaluation

To quantify how consistently the above two observed mechanisms appear across embedding dimensions, we define three metrics:

1. **Sorting accuracy:** For each possible initial permutation, we verify that the choice of transposition is a correct swap. Accuracy is defined as the fraction of correct swaps, which allows us to calculate an accuracy between 0 and 1. If an agent has an accuracy $= 1$, it will be able to sort any permutation optimally.
2. **Global order encoding:** Consider the attention weights $W = QK^\top/\sqrt{d_k}$, with final row W_ℓ. We compute the proportion of non-inversions between $(\pi(1), \ldots, \pi(n))$ and $(W_{\ell,1}, \ldots, W_{\ell,n})$, where an inversion is defined as a pair of elements which are out of order between the two sequences. We then normalize by $\binom{n}{2}$, the maximum number of inversions between two sequences. This yields a metric in $[0, 1]$, with 0 meaning the attention weights are completely out of order, and with 1 indicating perfect alignment.
3. **Difference-based swap rule:** The agent's choice of transposition corresponds to the largest (or smallest) difference in values. For every initial permutation, we sort the differences between consecutive attention output values. We then find the ranking of the transposition which was actually chosen according to this sorted list. We measure the frequency with which the chosen swap lies among the top-k predicted differences, so a top 2 proportion of 1 would mean all swaps chosen by the agent were in the top 2 (of n) predictions.

4.2 Implementation details

Agents were trained using PPO using a learning rate of 2.5e-4 and discount factor 0.99 (full hyperparameters are in Appendix 8.2) [21]. Training was performed for 1M, 2M, or 10M timesteps. All models were stripped down transformers with an embedding layer, a single-head self-attention block, and a linear layer. Training was performed on Nvidia H100 GPUs.

4.3 Experimental design

We trained agents with embedding dimensions varying from 2 to 128 and sequence lengths of 6 and 8. In total, 475 agents were trained, with multiple random seeds, enabling a robust estimate of both mean performance and variability, which we later connect to embedding dimension. Length 6 sequences have a relatively small permutation space (720 possible states), making convergence feasible for nearly all runs, while length 8 (40,320 possible states) introduces enough combinatorial growth without becoming computationally prohibitive.

Importantly, the environment only incentivizes agents to eventually sort a sequence, not to discover the *optimal* sorting path. The optimal solution is not unique, and suboptimal but valid sorting strategies may still achieve perfect reward. Our analysis therefore focuses on the internal mechanisms agents converge to, rather than on efficiency.

Embedding dimension was chosen as our primary independent variable because it directly controls the representational capacity of the attention mechanism. This choice also allowed us to probe the trade-off between minimal capacity sufficient for high performance and excess capacity that may encourage structured representations. In principle, embedding dimension directly modulates representational capacity. A larger query/key dimension enhances the expressive power of attention, allowing the

network to approximate complex functions [1]. Additionally, increasing embedding dimensionality enriches the granularity of token embeddings, which empirically improves generalization.

We chose PPO for policy optimization because of its balance between stability and sample efficiency; it avoids catastrophic policy updates via clipping while remaining easy to parallelize [8].

Notably, while we tested a large range of embedding dimensions, the results generally only changed below an embedding dimension of approximately 30 and leveled out after that. There were also some differences in results for the different sequence lengths which should be investigated further, but this was not the main focus of the experiment and would require much more computation because of the combinatorially increasing state space.

5 Results

We observed that agents consistently converged to a simple and interpretable algorithm when solving the sorting task. Specifically, two mechanisms emerge consistently across runs:

1. **A global order encoding in attention weights:** the final row of the attention weight matrix maps each number token to a value between 0 and 1, and lower attention weight values correspond to lower numerical tokens. This means that the model discovers the global ordering of the input tokens (Observation 1)
2. **A difference-based rule for swap selection:** The value matrix and final linear layer of the network calculate the difference between consecutive attention weight values, and the chosen transposition corresponds to the largest difference (Observation 2)

We hypothesize that together, these two observations define a simple circuit underlying sorting behavior. Importantly, while small embedding dimensions are sufficient for near-perfect accuracy, larger embedding dimensions make these mechanisms more consistent, more faithful, and more robust across agents. In the following subsections, we show evidence that agents reliably learn this circuit, and that convergence to it becomes more consistent as embedding dimension increases.

5.1 Accuracy of agents

Agents reliably learned to sort sequences under a wide range of embedding dimensions. For sequence length 6, 99.2% of the agents reliably achieved 100% accuracy once the embedding dimension was greater than 16. For the sequence length of 8, many agents still achieved 100% accuracy, but at a lower rate of 37.4% when the embedding dimension was above 16. Our observations relate to models which achieve a perfect or near-perfect accuracy, and the PPO training algorithm was able to achieve this for many agents at each length. The average accuracy at each embedding dimension is shown in Figure 1, which reflects these results. The average accuracy levels out for both sequence lengths at a low embedding dimension but is higher for length 6 sequences.

Notably, accuracy saturates at relatively low embedding dimensions, which suggests that task success requires only limited representational capacity. To understand what additional capacity provides, we next examine internal representations.

Table 1: Average accuracy of the agents by length and number of iterations.

	Length 6	Length 8
1M iters	**96.7%**	N/A
2M iters	N/A	74.3%
10M iters	95.5%	**90.7%**

5.2 Representation of global ordering

Even though agents only needed to predict single swaps, most developed a coherent representation of the *entire* sequence ordering in the last row of the attention matrix. We quantified this with the proportion of non-inversions (see metric 2).

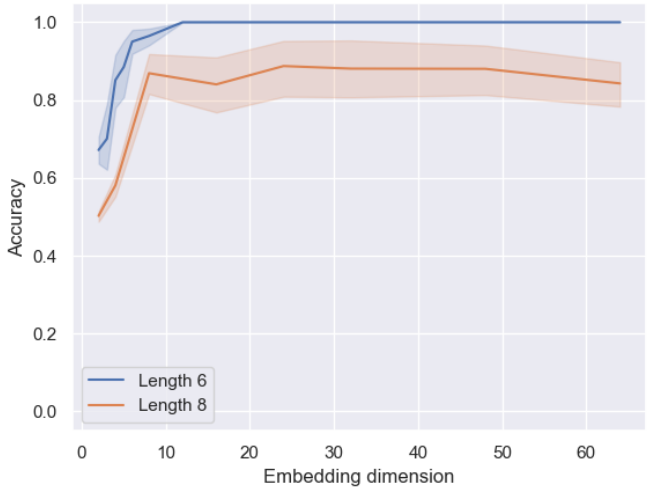

Figure 1: Accuracy vs. embedding dimension for length 6 and length 8 agents. Almost all agents achieve 100% accuracy for length 6, but this is not the case for length 8, where agents either achieve very high accuracy or get stuck at much lower accuracy. The mean and 95% confidence intervals are shown.

For agents trained on sequences of length 6, the average proportion of non-inversions increases as the embedding dimension increases but levels out at 87% around embedding dimension of 30. This means there are on average only around 2 inversions between the input ordering and the attention output ordering.

For agents trained on sequences of length 8, the trend still exists, but it is not as strong because some agents never converge on a solution with 100% accuracy. If these agents are ignored, we see a similar result to the agents trained on length 6 sequences, where the average proportion of non-inversions increases until an embedding dimension of around 30 at which point it levels out at 78%. The agents with a lower accuracy only reach an average of 57%, close the 50% which would be expected for a random sequence. Figure 2 shows this trend, where the proportion of non-inversions is close to 50% at a low embedding dimension and increases as embedding dimension increases until it levels out, providing evidence for Observation 1 (see Appendix 8.3 for additional visualizations of attention weights). Crucially, this metric increases well after the accuracy has already saturated. Larger embeddings therefore make its internal ordering representation stronger and more reliable.

5.3 Decision mechanism

We also found strong evidence for Observation 2: agents typically selected swaps by identifying the most positive or most negative difference between consecutive values in the last attention row. We analyzed how often the move an agent choose to make was in the top i predictions according to the largest gaps in the attention matrix (metric 3). Examples of attention weight visualizations for high- and low-performing agents are shown in Appendix 8.3. Across high-accuracy agents, 76–77% of moves matched the top-1 prediction, and more than 90% matched the top-2 (Table 2, Figure 3).

As with ordering fidelity, swap prediction alignment seemed to increase with embedding dimension until leveling off at around dimension 30. This shows that higher embedding dimensions not only strengthen global representations but also sharpen the decision rule based on them.

5.4 Failure modes

Beyond aggregate accuracy metrics, we examined qualitative failure modes of low-dimension agents (Appendix 8.3 includes representative training loss curves). A common pattern was what we term the "local greedy trap": swapping the most obviously incorrect local pair due to a failure to recognize

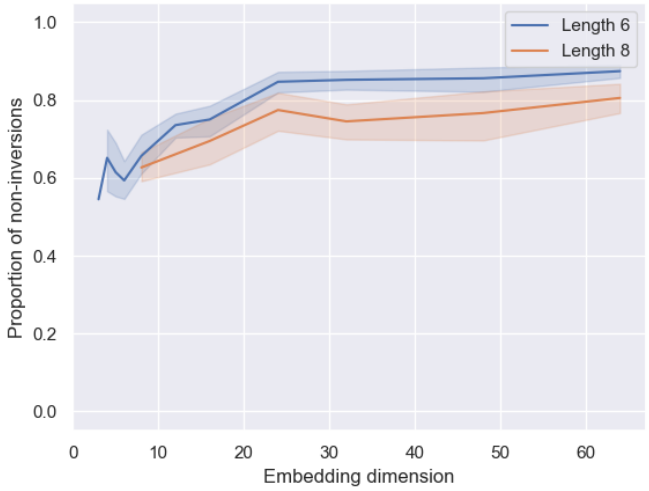

Figure 2: Proportion of non-inversions vs. embedding dimension for length 6 and length 8 sequences where the model has near-perfect accuracy. This reaches over 80% on average for both sequences when the embedding dimension is large enough. The mean and 95% confidence intervals are shown.

that fixing a larger global inversion sometimes required temporarily increasing the number of local inversions. This trap is especially telling because it indicates that the model had not formed a coherent global ordering representation; instead, it relied on a heuristic driven purely by local differences in value embeddings. The appearance of this failure mode decreases with higher embedding dimensions, suggesting that greater representational capacity allows simultaneous encoding of both global and local order features.

Table 2: The proportion of moves an agent with high accuracy chooses which are the first/second largest gap in the last row of the attention matrix.

	Top 1	Top 2
Length 6	76.2%	**94.0%**
Length 8	**76.8%**	92.5%

This table includes almost all agents trained on length 6 sequences because they are almost all fully accurate. However, many length 8 agents never reach full accuracy so they are left out. The low-accuracy length 8 agents only have a top-2 proportion of 54.1%, which is much lower than the accurate agents. Figure 3 shows the proportion of correct top 1 and top 2 predictions as embedding dimension changes for the high accuracy agents. Similar to the proportion of non-inversions, these proportions increase with the embedding dimension until they level out at an embedding dimension of around 30.

5.5 Cross-metric consistency

Our two observations for how the model should act are very related. If the global ordering observation is true, then choosing the move with the most negative difference will always lead to a correct swap. Because of this, it makes sense that the performance of these two mechanisms against the embedding dimension are strongly related. This relationship can be seen in Figure 4, where there is a fairly strong association between the metrics used to evaluate Observation 1 and Observation 2. This suggests that either metric could serve as a proxy for the other in similar studies.

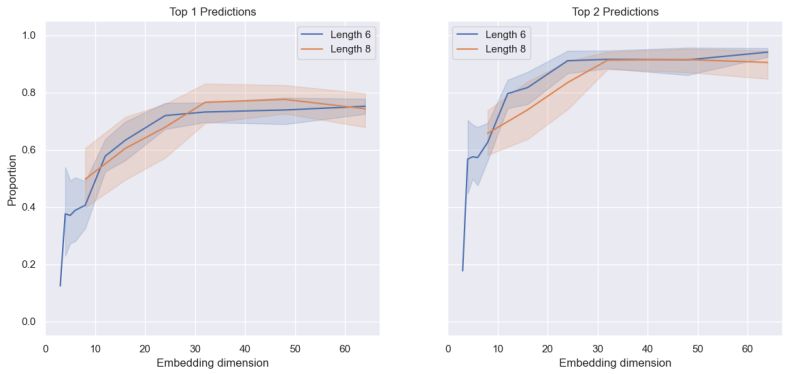

Figure 3: Proportion of moves which are the top 1/2 prediction according to our observation vs. embedding dimension for length 6 and length 8 sequences where the model has near-perfect accuracy. Almost all moves are chosen according to this observation, with the top 2 proportion reaching above 90% for a large enough embedding dimension. The mean and 95% confidence intervals are shown.

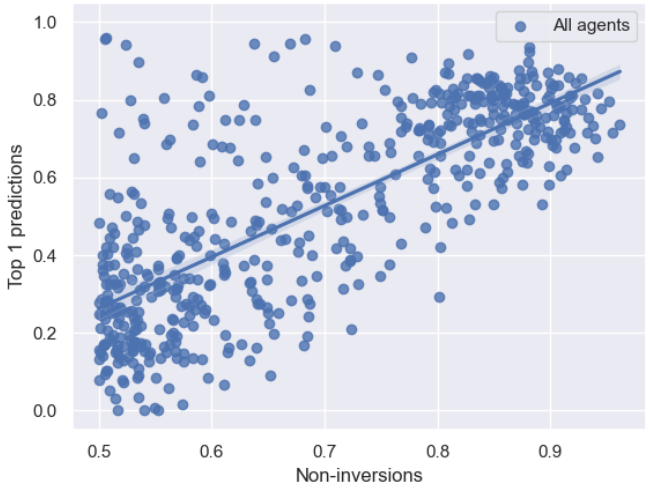

Figure 4: The proportion of top 1 predictions vs. proportion of non-inversions for all agents. These two metrics have an r^2 value of 0.56, so there is a fairly strong correlation between the two values. The line of best fit and 95% confidence interval for the parameters of this line are shown.

6 Discussion

The observation that representation quality continues to improve well after accuracy saturates has several implications. First, it highlights the importance of probing internal representations directly rather than relying solely on performance metrics, especially when evaluating model capacity. In safety-critical or interpretability-focused contexts, a more structured internal representation may be preferable even if accuracy is unchanged, as it can make the model's decision process more predictable and transparent.

From a scaling perspective, our findings parallel the idea of "capability plateaus" in large-scale models: increases in size yield diminishing returns on benchmark scores but continue to shape the internal geometry of the learned representation space. This reshaping may have downstream benefits, such as improved transfer to longer sequences, greater robustness to noise, or better alignment with human-interpretable concepts. In the RL setting, it could also reduce policy brittleness, as more structured latent states may generalize more smoothly to out-of-distribution configurations.

Finally, the fact that higher capacity leads to more consistent circuit formation across seeds hints at a practical trade-off: overprovisioning model width might be a cheap way to make learned policies more interpretable without any loss in performance, even if such overprovisioning would be avoided in a strict efficiency optimization.

The main observation from our results is that the accuracy of the models levels out at very close to 100% at a very low embedding dimension (around 6 for length 6 sequences trained for 10M timesteps), but the metrics discussed above increase well beyond that (around 30 for the same agents). This means that agents are learning how to solve the given problem perfectly with even a low embedding dimension, but are more likely to learn our simple interpretable solution as the embedding dimension keeps increasing.

Another result which gives insight into how the agents learn is that for the agents trained on length 8 sequences, not every agent learned our observations but the best agents almost all did. Some agents never achieved an accuracy over 90% even after 10M iterations of training. None of these agents performed very well on our metrics for evaluating how close an agent was to satisfying our observations, but there was a noticeable improvement among the agents which achieved near perfect accuracy. This means that in our experiments for the length 8 sequences, being closer to matching our observations meant that the agent generally performed better. Our observations for how the agents would solve the problem are, in some sense, the easiest solution they could learn.

7 Conclusion

We studied how embedding dimension affects the emergence of interpretable internal mechanisms in small transformers trained with PPO to perform bubble-sort–style adjacent swaps. Across hundreds of agents, we found two consistent patterns: the final row of the attention weights encodes a global ordering of tokens, and the chosen swap corresponds to the largest adjacent difference in that ordering. While accuracy saturates at relatively low embedding dimensions, the faithfulness and robustness of these mechanisms continue to improve up to dimension around 30. Thus, capacity improves representation quality, providing quantitative evidence that transformers build simple, structured world models even in constrained RL settings.

7.1 Applicability

Our analysis demonstrates that even very small transformers, when trained in RL settings, naturally converge to simple, interpretable circuits for solving sorting tasks. This supports the idea that transformers, under constrained conditions, expose internal mechanisms that are easy to study. Similar approaches could be applied to other algorithmic tasks (e.g., graph problems, dynamic programming), where the attention matrix may again reveal hidden internal structure.

7.2 Limitations

This study is limited both in scope and scale. We focused only on adjacent transposition sorting with 1–2 layer, single-head transformers. More complex architectures—such as deeper models, multi-head attention, or richer environments—were not explored. Computational budget also restricted our ability to test broader ranges of sequence lengths or larger training runs. As such, our findings should be viewed as evidence of a general trend, not as an exhaustive characterization.

7.3 Further work

Several extensions follow naturally. This paper examined how a transformer learned to execute an algorithm similar to bubble sort, where transpositions can only be made between two adjacent elements. This setup will be maximally efficient as long as correct swaps are always chosen, but there is room for efficiency improvements in the more general case. One direction is to relax the adjacency constraint and allow arbitrary in-place swaps; this could reveal whether transformers discover known efficient algorithms (e.g., merge sort) or converge to novel heuristics.

In addition to generalizing to non-adjacent swaps, several directions stand out. One is to introduce stochasticity into the environment, such as noisy token values or partial observability, to test whether the same ordering circuit emerges under uncertainty. Another is to explore hybrid training setups,

where a model is pre-trained in a supervised manner on optimal swaps before being fine-tuned with RL; this could reveal whether the ordering representation is more robust when acquired via imitation or exploration.

Further, introducing multi-head attention would allow us to examine whether different heads naturally specialize in complementary subproblems (e.g., local comparison vs. global structure). It would also be valuable to test the robustness of these circuits under pruning or quantization, which could preserve interpretability while reducing model size. Lastly, applying similar analyses to structurally richer algorithmic tasks, such as graph algorithms, constraint solvers, or an algebra problem, could test the generality of the capacity–representation link and might uncover new, reusable circuit motifs.

References

[1] Noah Amsel, Gilad Yehudai, and Joan Bruna. Quality over quantity in attention layers: When adding more heads hurts. In *The Thirteenth International Conference on Learning Representations*, 2024.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, pages 6243–6267. PMLR, 2023.

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *CoRR*, abs/2110.14168, 2021.

[5] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[6] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[7] Neetu Faujdar and Satya Prakash Ghrera. Analysis and testing of sorting algorithms on a standard dataset. In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 962–967. IEEE, 2015.

[8] Wenbin Guo, Zhao Li, Xin Wang, Zirui Chen, Jun Zhao, Jianxin Li, and Ye Yuan. Convd: Attention enhanced dynamic convolutional embeddings for knowledge graph completion. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[9] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

[10] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[11] Steffen Herbold. Sortbench: Benchmarking llms based on their ability to sort lists. *arXiv preprint arXiv:2504.08312*, 2025.

[12] Zhuoqiao Hong, Haocheng Wang, Zaid Zada, Harshvardhan Gazula, David Turner, Bobbi Aubrey, Leonard Niekerken, Werner Doyle, Sasha Devore, Patricia Dugan, et al. Scale matters: Large language models with billions (rather than millions) of parameters better match neural representations of natural language. *BioRxiv*, 2024.

[13] Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. Making large language models a better foundation for dense retrieval. *CoRR*, abs/2312.15503, 2023.

[14] Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

[15] Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, A. Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin A. Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nat.*, 618(7964):257–263, 2023.

[16] Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. The role of context types and dimensionality in learning word embeddings. *arXiv preprint arXiv:1601.00893*, 2016.

[17] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[19] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[20] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[24] Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. *Advances in neural information processing systems*, 31, 2018.

[25] Shizhuo Dylan Zhang, Curt Tigges, Stella Biderman, Maxim Raginsky, and Talia Ringer. Can transformers learn to solve problems recursively? *arXiv preprint arXiv:2305.14699*, 2023.

[26] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:27147–27166, 2023.

8 Appendix

8.1 Model

The transformer architecture was first introduced in the paper *Attention Is All You Need* and was able to achieve state of the art results for translation tasks [22]. Since then, transformers have been applied to a wide range of machine learning tasks across language, vision, and other domains. The core mechanism in the architecture is the attention function, which calculates output vectors from input queries, keys, and values. In self-attention, the queries, keys, and values are all calculated from the same input sequence.

The model used in this experiment is a stripped down transformer with an embedding layer, a single-head self-attention block, and a linear layer. The toy problem we considered was extremely simple and could conceivably be solved with 100% accuracy by almost any sufficiently complex model. Our goal was to isolate attention weights to gain insight about how the problem was solved, so we took away as much of the rest of the model as possible. The main parameter of the model which was controlled to see how the results changed was the embedding dimension, which was also equal to the dimension of the keys/queries.

8.2 Training algorithm

PPO is a reinforcement learning algorithm which is simpler and more performant than many other leading algorithms. It does this by optimizing a lower bound of the policy performance created from clipped probability ratios. This can be implemented with an Actor-Critic framework, where the actor chooses actions and the critic estimates the value function. The algorithm alternates between running the policy and optimizing for multiple epochs based on the results [21].

Our model is given a fixed-length permutation as a series of tokens (with no predefined numerical value) and output a single index representing a transposition which should make the input permutation closer to being sorted. This is trained using the PPO reinforcement learning algorithm [21] and a Gym environment [2]. The agent receives a reward of 1 if the permutation is sorted after making a transposition and a reward of -0.001 otherwise. The initial permutation is randomized and each run terminates when either the permutation is fully sorted or 1000 transpositions have been made. The clipped value loss of the PPO algorithm [21] during an example training run can be seen in Figure 8.

Table 3: PPO parameters used while training the agents.

Parameter	**Value**
Max steps for Gym environment	1000
Total timesteps	1M/2M/10M
Learning rate	2.5e-4
Number of environments	8
Number of steps per policy rollout	128
Discount factor	0.99
General advantage estimation lambda	0.95
Number of minibatches	4
Surrogate clipping coefficient	0.1
Entropy coefficient	0.01
Value function coefficient	0.5
Maximum norm for gradient clipping	0.5

8.3 Additional figures

One of our main observations from this paper is that the final row of the attention weight matrix represents the global ordering of the input sequence. Figure 5 shows an agent's attention weight matrix when the input is fully ordered. If Observation 1 held, the last row of this matrix would contain fully ordered elements, which is exactly what happens.

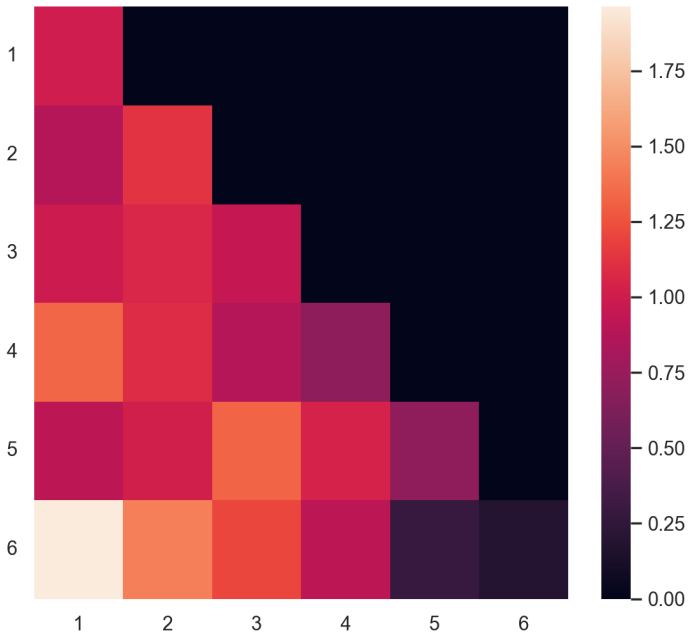

Figure 5: Visualization of the attention weights for a fully ordered permutation (using the same agent as in Figure 6). The weights in the last row are ordered just like the permutation, showing that for this agent and permutation there are no inversions.

This same phenomenon can also be visualized by looking at the spread of attention weight values corresponding to each token in the input sequence. Figures 6 and 7 show this for two example agents. Each figure shows a violin plot of the attention weight values of every token. For example, the violin above 2 shows the spread of last-row attention weights of the agent corresponding to the column where the input token was 2. A wider violin corresponds to a higher density of values. Both Figure 6 and Figure 7 show monotonic averages of the attention values, which is exactly what we would expect to see if the agent learned the global ordering of the tokens. It is important to note again that the model is never given the numerical values of the tokens, but is able to figure out their relative ordering just from the training process (with a large enough embedding dimension). Figure 7 has a much wider spread of values for each token, and this increased variation explains why it gets the ordering wrong more often.

The clipped value loss of an example training of an agent is shown in Figure 8. This is the loss for an agent which was able to achieve 100% accuracy, so the loss converges to 0. For agents which never converge to a fully accurate solution, the loss would likely not stabilize. This happens more when the sequence length is 8 because the search space to find correct solutions is so much larger.

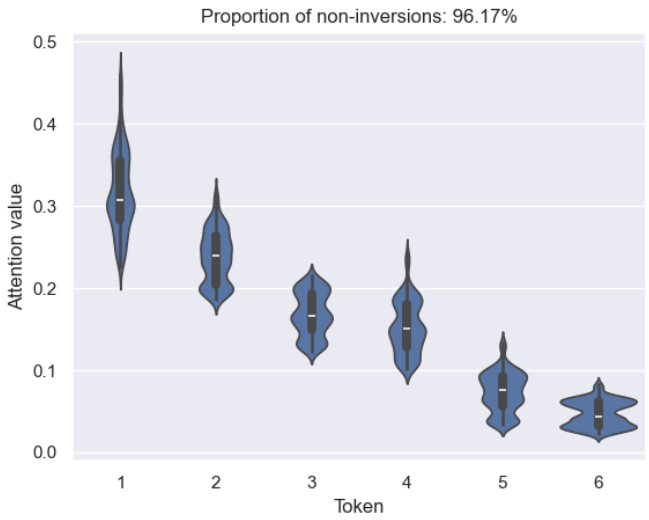

Figure 6: Attention weights by token for the agent with fewest average inversions.

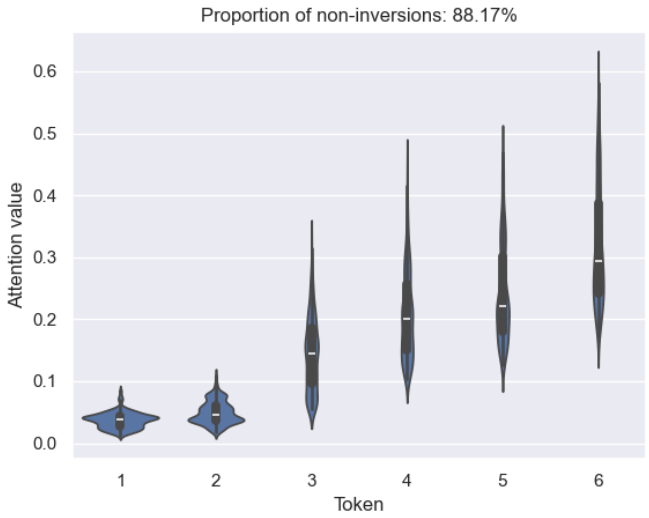

Figure 7: Attention weights by token for a random agent with high accuracy.

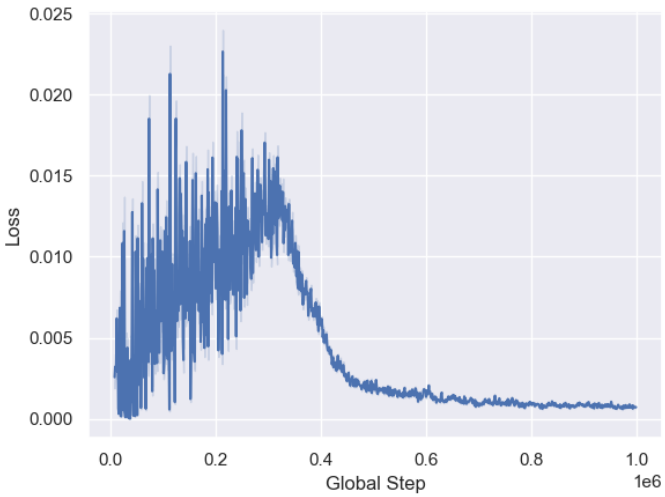

Figure 8: Clipped value loss vs. global step in an example training of an agent on a length 6 sequence.