Scalable Neurosymbolic LLM Reasoning via Multi-Token Encoding

Anonymous Author(s)

Abstract

Large language models (LLMs) struggle with reliable rule-based reasoning, particularly in mathematical domains requiring precise computation. Recent neurosymbolic approaches have shown promise by encoding LLM hidden states into neurosymbolic representations to perform rule-based reasoning, but current methods rely on linear encoders applied to a single token's hidden state, limiting representational capacity and robustness to input variations. This work extends previous neurosymbolic LLM methods by replacing linear encoders with transformer-based architectures that encode multiple hidden states from the model's forward pass, enabling the model to capture information distributed across longer input sequences. We evaluate the method on arithmetic prompts in a low-data 3-digit setting with varied question formats and in 4/6-digit scaling experiments. The transformer encoder achieves 93.1% average accuracy, compared to 53.6% for the linear encoder, and remains substantially more robust as input length increases, avoiding the problem-type identification failures that cause the linear encoder to collapse on longer inputs.

1 Introduction

While large language models (LLMs) have achieved unprecedented success across numerous domains, they face fundamental challenges when tackling problems that demand precise computational accuracy and strict adherence to logical rules [Hendrycks *et al.*, 2021; Cobbe *et al.*, 2021]. LLMs demonstrate exceptional ability in recognizing patterns and generating contextually appropriate responses by learning from massive datasets, yet their probabilistic nature becomes a liability in mathematical reasoning tasks where exactness is required. LLMs frequently produce inconsistent outputs and erroneous calculations when confronted with rule-based problems, as their training does not necessarily prioritize computational correctness, but instead optimizes for plausible generation based on token sequences seen during training [Cobbe *et al.*, 2021].

The core issue lies in their reliance on statistical patterns rather than systematic algorithmic execution, leading to unreliable performance in domains where precision cannot be compromised. Conversely, traditional symbolic computation systems excel precisely where neural models falter, since they guarantee accurate results through deterministic rule application and logical inference. However, symbolic approaches face their own limitations: they lack the adaptability and contextual understanding that neural models provide, making them brittle when confronted with the ambiguity and variability inherent in natural language inputs. This creates a fundamental trade-off between the adaptability and robustness of neural systems and the computational reliability of symbolic methods, thereby motivating the development of hybrid approaches that can bridge this gap [Garcez and Lamb, 2023].

The Neurosymbolic-LLM framework of Dhanraj and Eliasmith addresses this problem by encoding LLM hidden states into vector symbolic algebra (VSA) representations [Plate, 1995; Kanerva, 1988; Gayler, 2004; Levy and Gayler, 2008; Schlegel *et al.*, 2022], applying symbolic algorithms to the extracted representation, and decoding the resulting symbolic solution back into the LLM's forward pass. However, the original approach uses a linear encoder applied only to the hidden state of the most recent token. The original paper identifies this as a limitation and direction for future work, since a linear encoder creates a bottleneck: the encoder can succeed only when the final-token representation contains enough information to recover the full symbolic structure of the input. As prompts become more varied or numerical inputs become longer, the relevant information is more likely to be distributed across multiple token representations. This concern is consistent with probing work showing that the recoverability of linguistic and structural information depends strongly on which hidden states are queried and how they are probed [Hewitt and Manning, 2019; Liu *et al.*, 2019].

In this work, we replace the single-token linear encoder with a transformer encoder [Vaswani *et al.*, 2017] that operates over the hidden states of all tokens in the input sequence. This allows the encoder to construct VSA representations from sequence-level information rather than relying on a single compressed final-token state. We evaluate this change in two settings: a low-data 3-digit arithmetic setting with varied question formats, and 4-digit and 6-digit scaling experiments designed to test whether multi-token encoding becomes more

important as relevant information is distributed across longer input sequences.

Our key contributions are (1) a transformer-based encoder architecture for VSA generation that leverages hidden states from the full input sequence, (2) an analysis of layer-wise encoding behaviour showing that the transformer-based encoder remains effective at earlier layers where the linear encoder struggles under limited data, and (3) a scaling study showing that the transformer-based encoder maintains stronger performance on 4-digit and 6-digit inputs, whereas the linear encoder largely fails as input length increases.

2 Related Work

2.1 Neurosymbolic-LLM

Recent work has explored integrating symbolic computation with large language models in order to improve performance on tasks that require exact rule following. The most directly relevant prior work for this paper is the Neurosymbolic-LLM framework of Dhanraj and Eliasmith, which converts hidden states of an LLM into Vector Symbolic Algebra (VSA) representations, applies a symbolic algorithm to the extracted representation, and decodes the symbolic result back into the model's forward pass. In the original formulation, the encoder is a single linear layer applied to the hidden state of the most recent token. This design proved effective on arithmetic reasoning, but it also makes symbolic extraction dependent on whether the final-token representation contains enough information about the full problem.

2.2 Interpretability

A large body of interpretability work studies what information is recoverable from LLM hidden states and how that information is organized across layers. Linear and structural probes test whether specific properties can be extracted from internal representations using simple supervised mappings [Hewitt and Manning, 2019; Liu *et al.*, 2019]. More recent mechanistic interpretability work has focused on higher-level features and circuits, showing that model behavior can often be better understood in terms of structured latent features rather than individual units [Elhage *et al.*, 2021; Lindsey *et al.*, 2025].

Sparse autoencoders (SAEs) and related dictionary-learning approaches provide one influential method for decomposing hidden states into sparse, more interpretable features [Bricken *et al.*, 2023; Gao *et al.*, 2024]. These methods have been useful for identifying candidate concepts, studying monosemanticity, and tracing which features are active during particular computations. However, the learned features are discovered in an unsupervised way and do not in general correspond to a predefined symbolic structure that can be directly manipulated by an external algorithm.

The neurosymbolic LLM framework proposed by Dhanraj and Eliasmith also aims to decompose hidden states into interpretable components, but instead learns to map hidden states into a structured VSA representation whose components correspond to explicit symbolic roles, values, and operations. Because the transformer-based encoder proposed in this paper produces substantially more accurate VSA representations than the original linear encoder and modifies that representation to steer the LLM, the resulting neurosymbolic decomposition provides a more accurate picture of which structured concepts the model uses, and thus furthers the interpretability of the model during this task.

3 Methodology

This section describes our extension to the neurosymbolic LLM framework proposed by Dhanraj and Eliasmith through the replacement of linear encoders with transformer-based architectures. We first analyze the limitations of linear encoding for VSA generation, then present our transformer encoder design and demonstrate its advantages for processing contextual information.

3.1 Design Motivation

The original neurosymbolic approach uses a linear encoder applied to the hidden state of the most recent token. This design is effective when the relevant problem information is already concentrated in the hidden state of the most recent token, but it becomes more restrictive when prompts are more variable or when longer numerical inputs distribute relevant information across a larger portion of the sequence. Our modification is therefore to replace the single-token encoder with a transformer-based encoder that operates over the hidden states of all tokens.

3.2 Data and Evaluation Settings

All experiments use LLaMA 3.1 8B as the frozen base LLM, with VSA dimension $v = 2048$. Following the original setup, this model has 4096-dimensional hidden states at each of its 32 layers [Grattafiori and others, 2024]. We record hidden states immediately before they are processed by the selected layer, preserving the model's internal representation at that stage before the layer update. Unless otherwise stated, encoders are trained with the Adam optimizer using batch size 16 and an initial learning rate of 10^{-3}, which is reduced during training. Reported results are from single runs.

For the low-data experiments, we use a 1,000 training and 100 testing example version of the Symbolic-Math dataset with multiple question formats per problem type. For example, squared modulo questions may appear as "{x}^2 mod {y}", "({x}^2) mod {y}", "What is {x} squared mod {y}?" , "Calculate {x}^2 mod {y}.", and "Compute {x} squared modulo {y}." These experiments use 3-digit inputs.

Separately, for the scaling experiments, we construct 4-digit and 6-digit variants of the dataset, each with 10,000 training and 1,000 testing examples. We use the larger training set in these experiments to separate effects of input length from the low-data overfitting regime studied in the 3-digit setting. Unless otherwise noted, these longer-input experiments use the same question-format variation as the 3-digit setting.

3.3 Transformer Encoder Architecture

To address the linear encoder limitations, we replace it with a transformer-based architecture that can process the hidden states of all tokens in the input sequence. Our transformer

encoder provides direct access to the full contextual information available at the selected LLM layer, enabling more robust VSA generation across diverse input formats.

Model Architecture

The transformer encoder consists of four main components that process the hidden states of all tokens in the input sequence. The architecture includes an input projection layer that maps LLM hidden states from dimension d to an internal hidden dimension of 512, followed by four transformer encoder layers with 8 attention heads each and feedforward dimensions of 2048. After transformer processing, the final token's representation is selected and projected to the VSA dimension v through an output projection layer. The model uses GELU activation, 0.1 dropout rate, and bfloat16 precision for computational efficiency. Note that the total number of trainable parameters in the transformer encoder is 9.4 million, which is similar to the 8.4 million trainable parameters in the linear encoder approach.

The transformer encoder processes a sequence of LLM hidden states $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the LLM hidden dimension. The processing is as follows: input projection to internal dimension, multi-head self-attention and feedforward processing through four transformer layers, final token selection, and output projection to VSA space $\mathbf{v} \in \mathbb{R}^{v}$.

3.4 Implications for VSA Generation

Because the transformer-based encoder receives the hidden states of the full input sequence, it can learn mappings to VSA space that depend on relationships among multiple tokens rather than on a single compressed representation. This makes it a natural candidate for handling both varied prompt forms and longer numerical inputs. The experiments below test whether this architectural change improves (i) data efficiency in the low-data setting and (ii) robustness as numerical input length increases.

We also evaluate problem-type identification by decoding the operation component of the predicted VSA and counting cases where the selected symbolic routine does not match the ground-truth operation.

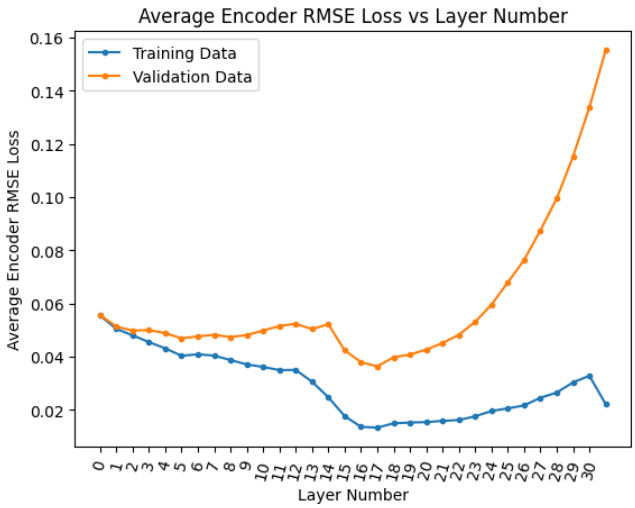
4 Results

4.1 Performance versus Intervention Layer

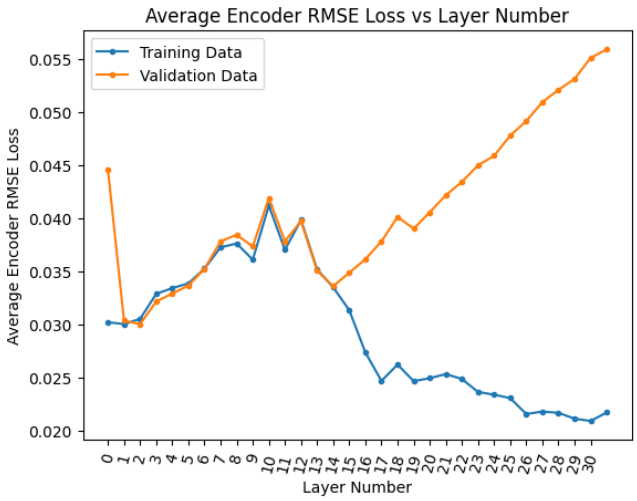
The choice of intervention layer significantly impacts encoder performance and generalization capabilities. Figure 1a shows the RMSE loss across all LLM layers for the linear encoder, while Figure 1b presents the corresponding results for the transformer encoder.

The linear encoder exhibits severe overfitting beyond layer 15-17. Training loss decreases steadily to approximately 0.014 at layer 17, while validation loss initially follows this trend but diverges after layer 17, reaching 0.16 by layer 30.

Note that this overfitting pattern only occurs when the training dataset size is limited to 1,000 training examples. When trained with 10,000 training examples, as in the original work by Dhanraj and Eliasmith, the encoders at each layer do not overfit, as shown in Appendix C. Additionally, this



(a) Linear Encoder



(b) Transformer Encoder

Figure 1: Encoder RMSE loss versus intervention layer for linear encoder (left) and transformer encoder (right). While both approaches result in overfitting in deeper layers of the LLM, the validation loss for the transformer-based encoder is much smaller at earlier layers, allowing for successful conversion of hidden states to VSAs by intervening earlier in the LLMs forward pass.

work introduces varied question formats whereas the original paper did not, which may increase the difficulty of encoding accurate VSA representations.

Similar to the linear encoder, the transformer encoder also overfits in later layers, as shown in Figure 1b. Unlike the linear encoder, however, validation RMSE loss of the transformer-based encoder is much lower at earlier layers. This means that the transformer based encoder can obtain accurate VSA representations on problems it has not seen during training by choosing the intervention layer to be earlier in the LLM's forward pass.

Using the layers at which each approach achieves a minimum validation RMSE score, we select the intervention layer

for the linear encoder to be 17 and the intervention layer for the transformer-based encoder to be 2. Selecting a fixed intervention layer for each approach is necessary in order to run the neurosymbolic system end-to-end in order to determine how the symbolic intervention improves the LLM's ability to output the correct answers.

4.2 End-to-End Performance

Table 1 presents the performance comparison across all problem types. The transformer encoder consistently matches or outperforms both the linear encoder and standard LLM across all problem types, achieving 93.1% average accuracy and 0.341 average cross-entropy loss compared to 53.6% accuracy and 2.389 loss for the linear encoder and 30.7% accuracy and 4.861 loss for the standard LLM. This corresponds to the transformer based encoder having on average a 73.7% higher accuracy and a 85.7% lower loss compared to the linear encoder, and a 203.3% higher accuracy and 92.9% lower loss compared to the standard LLM. In contrast, the linear encoder has on average a 74.6% higher accuracy and 50.9% lower loss than the standard LLM.

The substantial performance improvement of the transformer encoder over the linear encoder is particularly noteworthy given that both approaches use identical decoder architectures and symbolic algorithms. This suggests that encoding accuracy plays a pivotal role in overall neurosymbolic system performance, as the primary difference between the two approaches lies in the encoder's ability to generate accurate VSA representations from LLM hidden states.

The gating mechanism also helps preserve performance on problem types not used for symbolic training. On addition problems, the transformer encoder matches the standard LLM's perfect performance (100.0% accuracy), while the linear encoder achieves 95.7%. For integer division, the transformer encoder outperforms both the linear encoder (66.7% vs 50.0%) and standard LLM (52.8%), indicating effective generalization to problem types not encountered during training. The lower performance of the linear encoder on these problem types indicates the neurosymbolic system is incorrectly engaging for problem types not seen during training, which happens when the system cannot accurately determine the problem type asked in the prompt. The transformer-based encoder avoids this problem by more accurately encoding the correct problem type, leading to intervention only on problem types it has a symbolic algorithm for.

The linear encoder achieves higher accuracy than the standard LLM on all trained problem types except square modulo, where both approaches achieve 8.3% accuracy. This exception reflects the poor encoding accuracy of the linear encoder for this particular problem type, as shown in Appendix A.

4.3 Encoder Accuracy

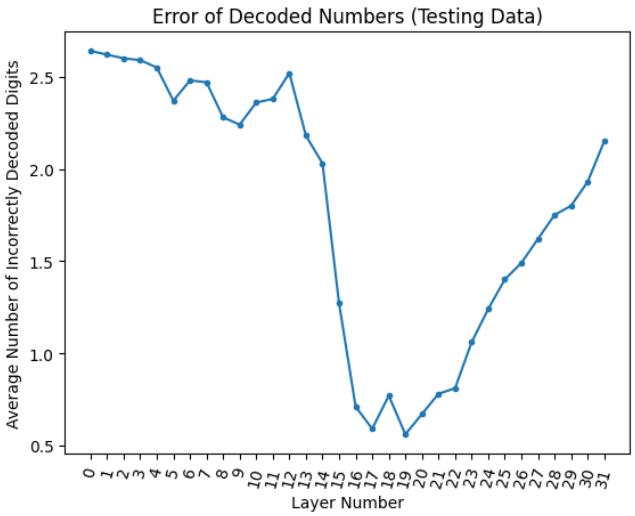
Figure 2a and figure 2b show the encoder average number of digits incorrectly represented per layer for the linear encoder and transformer-based encoder, respectively. Since 3 digit numbers are used in this problem, the maximum number of incorrectly encoded digits is 3, and the minimum number is 0. At their intervention layers (layer 17 for the linear encoder and layer 2 for the transformer encoder), the linear encoder achieves an error rate of 0.59 digits out of 3, and the transformer encoder achieves an error rate of 0.032 digits out of 3. This substantial difference in the encoding error rate helps explain why the transformer based model is able to more significantly boost the performance of the base LLM relative to the linear encoder method.

Table 1: Performance comparison across problem types. Scores are accuracies (%) and losses are cross-entropy values. Addition and Division are unseen during training.

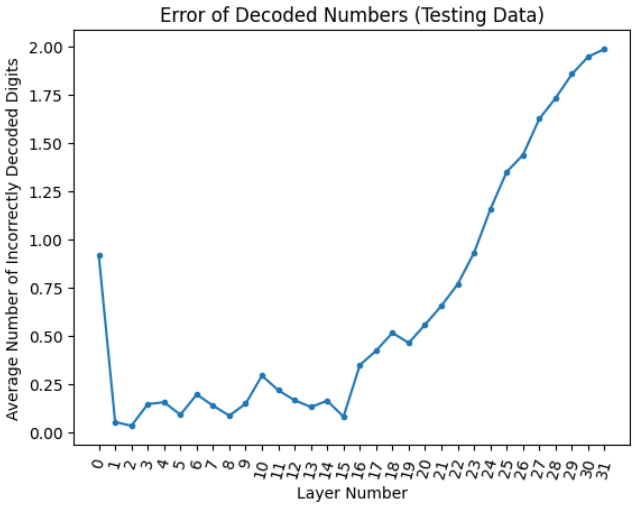
Problem	Model	Score ↑	Loss ↓
Mult.	Transformer	**100.0**	**0.032**
	Linear	23.8	3.942
	LLM	4.8	10.196
Modulo	Transformer	**87.0**	**1.258**
	Linear	69.6	1.458
	LLM	50.0	3.644
GCD	Transformer	**100.0**	**0.007**
	Linear	100.0	0.153
	LLM	73.1	1.131
LCM	Transformer	**100.0**	**0.051**
	Linear	24.1	4.201
	LLM	0.0	6.683
Sq. Mod	Transformer	**100.0**	**0.031**
	Linear	8.3	5.233
	LLM	8.3	4.973
AND	Transformer	**100.0**	**0.017**
	Linear	43.8	3.311
	LLM	3.1	6.064
XOR	Transformer	**100.0**	**0.050**
	Linear	75.0	1.954
	LLM	7.5	7.409
OR	Transformer	**76.9**	**1.851**
	Linear	46.2	3.161
	LLM	7.7	8.318
Add.	Transformer	**100.0**	**0.000**
	Linear	95.7	0.188
	LLM	**100.0**	**0.000**
Div.	Transformer	**66.7**	**0.108**
	Linear	50.0	0.186
	LLM	52.8	0.192
Avg.	Transformer	**93.1**	**0.341**
	Linear	53.6	2.389
	LLM	30.7	4.861

4.4 Scaling to Longer Numerical Inputs

To test whether the advantages of the transformer-based encoder extend beyond the 3-digit setting, we train both the linear and transformer-based encoders on 10,000 examples of 4-digit and 6-digit arithmetic problems, using the same varied question formats described in Section 3.2. We use 10,000 training examples in these scaling experiments to isolate the effect of input length from the low-data overfitting regime analyzed in the 3-digit experiments. Both encoders use layer 17 as the intervention layer in these experiments, matching the optimal layer identified for the linear encoder in prior work [Dhanraj and Eliasmith, 2025]. This provides a controlled comparison isolating the effect of input length: training data size, question format diversity, intervention layer, and decoder architecture are all held constant, with only encoder architecture and numerical input length varying across conditions. Note that alternative layer selections could potentially further improve the transformer-based encoder's results,



(a) Linear Encoder



(b) Transformer Encoder

Figure 2: Average number of incorrectly decoded digits versus intervention layer for linear encoder (left) and transformer encoder (right). The linear encoder achieves a minimum error rate around layers 17 and 19 of 0.5 digits out of 3, whereas the transformer-based encoder has a minimum error rate around layer 1 and 2 of 0.05 digits out of 3.

since it achieves low validation loss across a broad range of layers (see Appendix D).

As number length increases, the LLM tokenizer splits each number across more tokens, distributing the digit information across a longer subsequence of hidden states. This places a stronger burden on the linear encoder, which must recover all digit and operation information from the final token's hidden state alone, while the transformer-based encoder can attend directly over all token representations.

Table 2 summarizes performance on the eight *trained* symbolic task types: multiplication, modulo, gcd, lcm, square modulo, bitwise AND, bitwise XOR, and bitwise OR. In the

Table 2: Scaling comparison on longer numerical inputs, evaluated on the eight trained symbolic task types only. Misclassifications count incorrect problem-type identifications out of 1,000 evaluation questions.

Digits	Model	Acc. ↑	Loss ↓	Miscl. ↓
4	Transformer	**85.1**	**0.451**	**0**
	Linear	30.8	4.147	24
	LLM	12.3	10.718	–
6	Transformer	**68.9**	**1.860**	**0**
	Linear	10.4	4.732	45
	LLM	13.8	7.920	–

4-digit setting, the transformer-based encoder achieves an average accuracy of 85.1% and an average cross-entropy loss of 0.451, compared to 30.8% and 4.147 for the linear encoder, and 12.3% and 10.718 for the standard LLM. In the 6-digit setting, the transformer-based encoder continues to substantially outperform both baselines, achieving 68.9% average accuracy and 1.860 average loss, compared to 10.4% and 4.732 for the linear encoder, and 13.8% and 7.920 for the standard LLM.

From 4 digits to 6 digits, the average accuracy of the transformer-based encoder decreases by 16.2 percentage points (85.1% to 68.9%), whereas the linear encoder decreases by 20.4 percentage points (30.8% to 10.4%). More importantly, the transformer-based encoder remains substantially above both baselines at 6 digits, while the linear encoder largely collapses on most trained symbolic tasks. In the 6-digit setting, the linear encoder obtains 0.0% accuracy on multiplication, modulo, lcm, square modulo, bitwise AND, and bitwise OR, whereas the transformer-based encoder still achieves 80.5%, 55.0%, 84.0%, 49.5%, 60.0%, and 56.0% on these tasks, respectively.

4.5 Problem-Type Identification Under Increased Input Length

In addition to answer accuracy, we measured whether each encoder identified the correct problem type before symbolic intervention. This diagnostic is important because the neurosymbolic system depends on correct problem-type recognition in order to select the appropriate symbolic routine.

The transformer-based encoder produced 0 incorrect problem-type identifications in both the 4-digit and 6-digit settings. In contrast, the linear encoder produced 24 incorrect identifications in the 4-digit setting and 45 in the 6-digit setting. This shows that the degradation of the linear encoder on longer inputs is not limited to digit reconstruction. It also increasingly fails at the higher-level task of identifying which operation is being asked, which helps explain its sharp end-to-end performance drop in the scaling experiments.

5 Discussion

The experiments reveal two related limitations of the linear encoder. In the low-data 3-digit setting, it relies on later-layer final-token representations, where enough context has accumulated but the representations are more transformed and prone to overfitting. In the longer-input setting, the relevant

digit and operation information is distributed across more tokens, making the final-token bottleneck more severe. As a result, the linear encoder degrades sharply on 4- and 6-digit inputs, including 45 problem-type misclassifications out of 1,000 test examples in the 6-digit setting. The transformer-based encoder reduces both failure modes by directly integrating information across the full sequence of hidden states, achieving stronger early-layer encoding and zero problem-type misclassifications in both longer-input settings.

These results are consistent with the view that full-sequence access is an important contributor to the transformer encoder's advantage. The linear encoder depends on the LLM compressing all task-relevant information into a single token state before symbolic extraction, whereas the transformer encoder performs this integration explicitly. The enhanced performance of the transformer encoder suggests that the effectiveness of neurosymbolic intervention depends not only on the symbolic algorithm, but also on how structured information is extracted from neural representations before the symbolic computation is applied.

6 Limitations

Although this work tests inputs with increased variety and complexity relative to the original Neurosymbolic-LLM setting, the experiments remain restricted to arithmetic operations with predefined symbolic routines. In the current formulation, the encoder is trained in a supervised manner to map LLM hidden states onto known VSA targets, which assumes that the relevant symbolic structure is specified in advance. Extending this approach to more general reasoning domains may require moving toward unsupervised or self-supervised objectives that can independently discover structured neurosymbolic representations from model activations, analogous to how SAEs are trained.

A second limitation is that further experiments are required to determine which aspect of the transformer encoder is responsible for the observed improvement. For example, an ablation study with a nonlinear single-token encoder would help determine whether the gains come primarily from nonlinearity rather than sequence-level processing, while a sequence-level encoder without self-attention could help isolate the contribution of attention specifically.

7 Conclusion

We introduced a transformer-based encoder for neurosymbolic LLM reasoning that replaces single-token linear encoding with sequence-level encoding over all token hidden states. Across low-data, varied-prompt, and longer-input arithmetic settings, this architectural change improves robustness and scaling relative to the linear encoder. Future work should test whether the same advantage extends beyond arithmetic and whether multi-layer decoder interventions can further close the gap between accurate symbolic extraction and end-to-end generation.

References

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Varun Dhanraj and Chris Eliasmith. Improving rule-based reasoning in LLMs using neurosymbolic representations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30589–30608, Suzhou, China, November 2025. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024.

Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56:12387–12406, 2023.

Ross W. Gayler. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience, 2004.

Aaron Grattafiori et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *NeurIPS*, 2021.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4129–4138. Association for Computational Linguistics, 2019.

Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

Simon D. Levy and Ross W. Gayler. Vector symbolic architectures: A new building material for artificial general intelligence. In *Proceedings of the 2008 Conference on*

Artificial General Intelligence, pages 409–413. IOS Press, 2008.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1073–1094. Association for Computational Linguistics, 2019.

Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

Kenny Schlegel, Peer Neubert, and Peter Protzel. A comparison of vector symbolic architectures. *Artificial Intelligence Review*, 55(6):4523–4555, 2022.

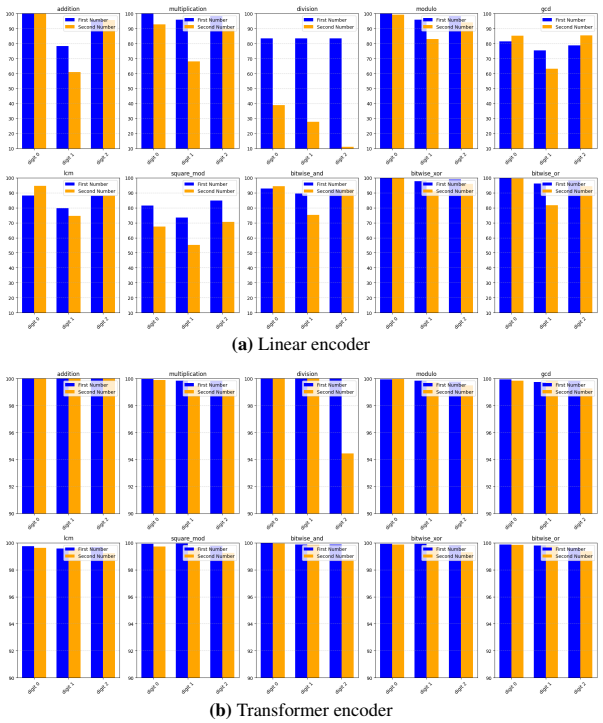
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

A Encoding Accuracy of Different Problem Types

The results in table 1 show that the linear encoder approach outperforms the standard LLM on most problem types, with the exception of the square mod, addition, and integer division problem types. In order to explain this phenomena, we can investigate the linear encoder's average encoding accuracy per problem type, as shown in Figure 3a. This plot shows that the encoding accuracies for the aforementioned problem types the linear encoder performed the worst on, the encoding accuracies are all on average lower than that of the other problem types.

In contrast the transformer encoder's average encoding accuracy per problem type, shown in Figure 3b, indicate that the average score for square mod, addition, and integer division problem types is not significantly lower than that of the other problem types. The average encoder accuracy for all problem types, including those not seen during training such as addition and integer division, is above 90%, which may partially explain why the transformer encoder was significantly more accurate than the linear encoder.

In addition to the encoding accuracy, there may be other reasons why the performance of both the linear encoder and transformer encoder are not 100%. For example, the transformer encoder has an accuracy over 99% for the modulo problem type, but the end-to-end performance for this problem type is only 87.0%.



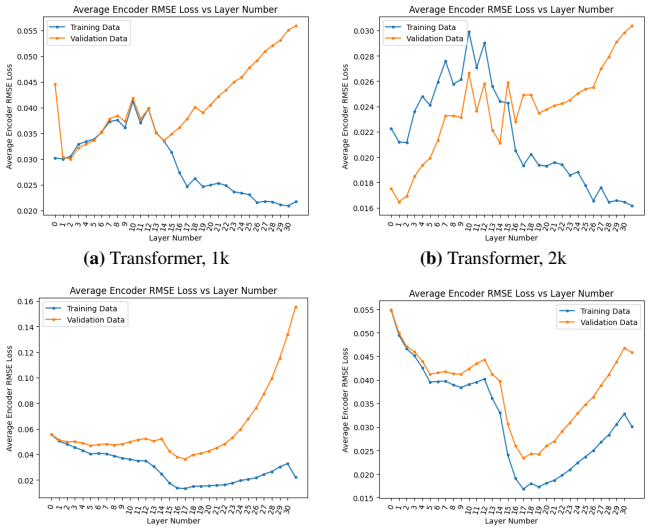
(a) Linear encoder

(b) Transformer encoder

Figure 3: Encoding accuracy per digit for each problem type. Integer division and addition were not included as training problem types.

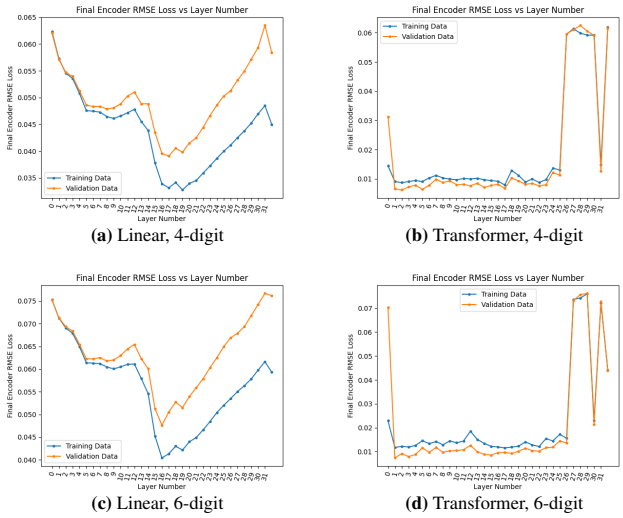
One potential reason for this phenomena is that for the modulo problem type, the algorithm the LLM is running in it's forward pass is difficult to steer just by changing the hidden state of the most recent token at a particular layer. In the case of the transformer encoder, it is attempting to change the most recent token's hidden state at layer 2. One way to analyze the algorithm an LLM is running is by creating an attribution graph, which is derived from an LLM by creating a cross-layer transcoder model, trained to predict the hidden state of future layers of an LLM using sparse activations in earlier layers Lindsey *et al.* [2025]. These attribution graphs show which sparse activations, or concepts, light up at various points in an LLMs computation process. For example, during addition, in earlier layers concepts related to the ones and tens digit of the earlier number may light up, and in later layers, concepts related to the sum of those two numbers may light up Lindsey *et al.* [2025].

Since different problem types have different attribution graphs, it is possible that changing the hidden state at a single layer for each problem type will be completely effective in raising the end-to-end accuracy to 100%. One possible solution would be to place a decoder layer at various layers in the LLM (each using the same solution VSA as input), in order to modify the hidden state of the model at different points of its forward pass. This was outside the scope of this current project, however, since it would require extra compute to train these additional decoder layers.



(a) Transformer, 1k **(b)** Transformer, 2k **(c)** Linear, 1k **(d)** Linear, 10k

Figure 4: Encoder RMSE loss versus intervention layer under different training set sizes.



(a) Linear, 4-digit **(b)** Transformer, 4-digit **(c)** Linear, 6-digit **(d)** Transformer, 6-digit

Figure 5: Layer-wise encoder RMSE loss for 4-digit and 6-digit inputs.

B Validation Loss of Transformer Encoder with 2,000 Rows of Data

Since Figure 2b showed that the transformer based approach overfit to the training data beyond a particular layer number, it is important to determine whether this behavior is reduced as the model is exposed to more training data. To test this, we train a transformer-based encoder on 2,000 rows of data, to see if the overfitting is reduced. Figure 4 shows the encoder validation and training loss of both the 1,000 and 2,000 row trained models. Note that the validation loss of the model trained on 2,000 rows of data is significantly less than that of the model trained on 1,000 rows of data, indicating that the overfitting behavior reduces as the training data size increase.

In the original work by Dhanraj and Eliasmith, the linear encoder was trained with 10,000 examples. The results in this section show that increasing the dataset size of the transformer encoder reduces the effect of overfitting, which suggests that a transformer-based encoder trained on larger amounts of data would be able to accurately extract VSA representations from intermediate layers in addition to early layers. This is consistent with the scaling experiments in Section 4.4, where the transformer-based encoder trained on 10,000 examples maintains low validation loss across a broad range of layers.

C Validation Loss of Linear Encoder with 10,000 Rows of Data

The results in section 4.1 indicate that the linear encoder model tended to drastically overfit when trained on 1,000 rows of data. In the original work by Dhanraj and Eliasmith, the authors also used a linear encoder, but it was trained with 10,000 rows of data. Figure 4 shows the impact of increasing the training dataset size to 10,000 rows from the 1,000 rows used in this study. Here, it is clear that the overfitting effect in later layers is much less significant when trained with 10,000 rows of data than when trained with only 1,000. This result indicates why Dhanraj and Eliasmith were able to accurately generate VSA representations from layer 16 of the LLM, since they used the larger 10,000 prompts to train their encoder.

D Layer-Wise Encoder Loss for 4-Digit and 6-Digit Inputs

Figure 5 shows the encoder RMSE loss versus intervention layer for both the linear and transformer-based encoders trained on 4-digit and 6-digit inputs, respectively. In both settings, the transformer-based encoder achieves substantially lower validation loss across the majority of layers compared to the linear encoder. Notably, the transformer-based encoder maintains low and stable validation loss across a wide range of early-to-mid layers in the 4-digit setting, whereas in the 6-digit setting the training and validation losses begin to diverge at later layers, indicating that even the transformer-based encoder faces increased difficulty in extracting accurate VSA representations from deeper layers as input complexity grows.

The linear encoder shows a consistently higher loss floor across all layers for both 4-digit and 6-digit inputs compared to the 3-digit setting (Figure 1), confirming that the single-token representational bottleneck becomes increasingly limiting as numerical input length increases.