TOWARDS INTERPRETABLE AND INFERENCE-OPTIMAL COT REASONING WITH SPARSE AUTOENCODER-GUIDED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel method that leverages sparse autoencoders (SAEs) and clustering techniques to analyze the internal token representations of large language models (LLMs) and guide generations in mathematical reasoning tasks. Our approach first trains an SAE to generate sparse vector representations for training tokens, then applies k-means clustering to construct a graph where vertices represent token clusters and weighted edges capture sequential token transitions. Using this graph, we define an edge-weight based reward function to quantify adherence to established reasoning traces, thereby identifying exploitative reasoning trajectories. Additionally, we measure generation diversity from clustering to assess the extent of exploration. Our findings indicate that balancing both exploitation and exploration is crucial for achieving high accuracy in mathematical reasoning tasks. During generation, the SAE can serve as a scalable reward model to guide generations, ensuring a balanced trade-off between exploitation and exploration. This prevents extreme behaviors in either direction, ultimately fostering a higher-quality reasoning process in LLMs.

1 INTRODUCTION

Recent advancements in language model training show new opportunities beyond the pre-training scaling law with model size, compute and dataset size (Kaplan et al., 2020). Inference-time scaling becomes increasingly important as reinforcement learning (RL) is incorporated into training pipelines for industry-grade LLMs (Guo et al., 2025; OpenAI, 2024b;a; AI, 2025). With RL, the models learn to explore rewards and new trajectories for problem solving that have never appeared in pre-training corpus (OpenAI, 2024a). LLMs trained with such inference-time search capability have demonstrated superior performance in reasoning benchmarks (Snell et al., 2024; Wu et al., 2024).

On the other hand, as the number of inference tokens scales, it becomes more and more challenging to track rewards for intermediate steps. Common approaches employed to track rewards include string matching (Wei et al., 2022) and using reward models (Ma et al., 2023), but they are not scalable as the search space becomes larger for harder tasks. Manual inspection is also unscalable, given the amount of work required.

As a result, there is a call for a mechanistic technique to supervise the generation process and assign appropriate rewards for intermediate search steps. An automated supervision pipeline for longCoT generation is useful not only for generation but also for improving RL training itself.

At inference time, recent research has shown that there is excessive redundancy in longCoT after training with long Chain-of-Thought (CoT) data obtained from an SFT checkpoint (Wang et al., 2025). As a result, a mechanistic technique that assigns rewards to guide model generation in more promising directions has great potential to save more tokens and improve generation efficiency.

In training, despite emerging techniques such as adding a length penalty to training loss or adding another stage of long2short RL training (AI, 2025), RL-based approaches remain costly. A mechanistic technique that ranks trajectories based on their balance between exploitation (shorter generations for promising directions) and exploration (longer generations with more reversions) during

RL enables methods like prioritized experience replay (Liu et al., 2023), facilitating more efficient learning and potentially leading to more optimal generations during inference.

In this paper, we present an unsupervised learning technique that uses the sparse autoencoder (SAE), which has been shown to learn interpretable representations (Gao et al., 2024) to generate clustering and construct a prior knowledge graph by learning from an existing data set that contains correct mathematical reasoning trajectories (Beeching et al., 2024). We first train an SAE to generate sparse vector representations of reasoning trajectories in the training data. Then, we apply k-means clustering to construct a reference graph, where vertices represent token clusters that capture conceptual information, and edges encode probabilistic transitions between them.

Using this graph, we establish a reference framework to evaluate how closely a reasoning trajectory aligns with existing trajectories in the training corpus by tracking the nodes and edges it traverses. We also investigate various metrics to quantify cluster diversity across different reasoning trajectories, capturing the extent of exploration. Our findings highlight a striking balance between exploitation and exploration, and considering multiple exploration metrics are important to achieving strong performance in mathematical reasoning tasks. This work demonstrates the potential of leveraging rich representations extracted from SAEs and employing an unsupervised learning approach as a highly scalable reward model to guide high-quality CoT generation.

2 Related Work

2.1 Sparse autoencoders

Sparse autoencoders (SAEs) (Makhzani & Frey, 2013) have emerged as a powerful tool for disentangling and interpreting internal representations in large language models (LLMs) (Cunningham et al., 2023; Bricken et al., 2023; Elhage et al., 2022). These models map high-dimensional hidden states into sparsely activated feature spaces, enabling more precise control and interpretation of model behavior.

Recent work has focused on improving SAE scalability and training stability. Gao et al. (2024) introduced TopK activation functions to enforce sparsity constraints more effectively, demonstrating success even with large models like GPT-4. Rajamanoharan et al. (2024) proposed gated sparse autoencoders, separating active latent selection from activation magnitude estimation, leading to improved reconstruction-sparsity trade-offs and enabling training of extremely wide autoencoders with up to 16 million latents.

In the context of LLM interpretability, Cunningham et al. (2023) showed that SAEs can decompose activations into interpretable features corresponding to specific concepts, while Marks et al. (2024) extended this work to discover sparse feature circuits that reveal causal relationships within models. Building on these advances, recent work has focused on leveraging SAEs for controlled generation: Yang et al. (2025) developed LF-Steering to address semantic inconsistency, and Yin et al. (2024) introduced Feature-level Preference Optimization (FPO) to improve alignment efficiency through sparse feature manipulation.

2.2 Chain-of-Thought Reasoning in LLMs

Chain-of-thought (CoT) reasoning has emerged as a powerful approach for enhancing language models' reasoning capabilities. First introduced by Wei et al. (2022), CoT prompting demonstrated that large language models can break down complex problems into intermediate reasoning steps by augmenting few-shot exemplars with step-by-step rationales. This discovery sparked numerous innovations in the field - from zero-shot CoT prompting with simple instructions (Kojima et al., 2022) to program-guided reasoning like Program-of-Thought (Chen et al., 2022) and PAL (Gao et al., 2023). Recent work has explored more sophisticated structures beyond simple chains, such as Tree-of-Thoughts (Yao et al., 2024) enabling exploration of multiple reasoning paths, Graph-of-Thoughts (Besta et al., 2024) incorporating cycles and verification steps, and Self-Consistency (Wang et al., 2022) leveraging multiple reasoning trajectories. The field has also seen advances in making CoT more robust through verification and refinement (Madaan et al., 2024), knowledge enhancement (Wang et al., 2023), and efficient reasoning techniques (Chu et al., 2023; Bi et al., 2024). Most recently, research has revealed that CoT capabilities may be inherent in pre-trained

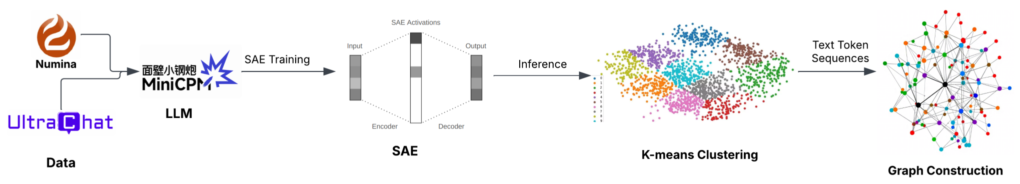


Figure 1: Flow of our pipeline

models even without explicit prompting (Wang & Zhou, 2024), suggesting these reasoning patterns emerge naturally during pre-training but require appropriate decoding strategies to surface.

Recent work has demonstrated the advantages of longer-form Chain-of-Thought reasoning through expanded context windows. Models like OpenAI's o1 (Jaech et al., 2024) and Kimi k1.5 (Team, 2025) show that scaling CoT to 128k tokens enables more comprehensive reasoning with improved performance on complex tasks. While these long-form approaches incorporate additional cognitive processes like planning and reflection, they face challenges of computational cost and potential ”overthinking” (Chen et al., 2024). This has motivated research into distilling long-form reasoning into efficient ”short-CoT” formats through techniques like preference optimization (Rafailov et al., 2024) and reinforcement learning-based generation (Ahmadian et al., 2024; Luo et al., 2025), balancing reasoning depth with practical constraints.

3 METHOD

This work proposes a method that leverages SAE and clustering techniques to analyze token representations for reasoning tasks. In this section, we provide a detailed description of our approach.

3.1 PRELIMINARY

Sparse Autoencoder Training. We first train a sparse autoencoder with a hidden dimension of $n = 8192$ à la Gao et al. (2024). Given an input token representation $\mathbf{x} \in \mathbb{R}^d$, the encoder maps $\mathbf{x}$ into a high-dimensional latent space as follows:

$$\mathbf{z} = \text{TopK}\left(W_{enc}\mathbf{x} + \mathbf{b}_{pre}\right), \tag{1}$$

where $W_{enc} \in \mathbb{R}^{n\times d}$ is the encoder weight matrix, $\mathbf{b}_{pre} \in \mathbb{R}^d$ is the pre-bias, and TopK denotes an activation function that only keeps the k largest latents, zeroing the rest.

The decoder then reconstructs the input as:

$$\hat{\mathbf{x}} = \left(W_{dec}\mathbf{z} + \mathbf{b}_{pre}\right), \tag{2}$$

where $W_{dec} \in \mathbb{R}^{d\times n}$ and $\mathbf{b}_{pre} \in \mathbb{R}^d$ are the decoder weight matrix and pre-bias, respectively.

The training objective is simply to minimize the reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \tag{3}$$

The training data consists of a balanced mixture (50/50) of samples from the NuminaMath (Beeching et al., 2024) and Ultrachat (Ding et al., 2023) datasets, applied to three MiniCPM (Hu et al., 2024) models (a 2B-parameter model, a 2B-parameter SFT model, and a 1B-parameter SFT model).

Sparse Representation Extraction. After training the SAE, inference is performed on 5,000 samples each from both the NuminaMath and Ultrachat datasets, corresponding to approximately 4 million tokens. For each token, the SAE outputs a list of the top activated latents together with their activation values.

Let $\mathcal{A}(\mathbf{x}) \subset \{1, 2, \ldots, 8192\}$ denote the set of neuron indices returned by the SAE for token $\mathbf{x}$, and let a_i denote the activation value corresponding to neuron $i \in \mathcal{A}(\mathbf{x})$. We then construct a sparse representation $\tilde{h}(\mathbf{x}) \in \mathbb{R}^{8192}$ by placing each retrieved activation at its corresponding index and

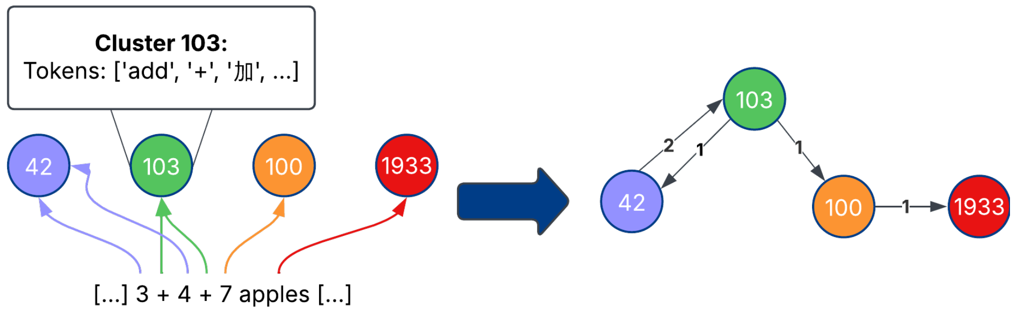


Figure 2: Sample construction of graph with the sentence "3 + 4 + 7 apples". Left shows cluster assignment of tokens. Right shows conversion to edges and nodes.

setting all other entries to zero:

$$\tilde{h}(\mathbf{x})_i = \begin{cases} a_i, & \text{if } i \in \mathcal{A}(\mathbf{x}), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

This sparse vector is subsequently used for downstream clustering and analysis.

3.2 Clustering via k-Means

To group semantically related tokens, we apply the k-means clustering algorithm to a randomly selected subset of the sparse vectors $\tilde{h}(\mathbf{x})_i$ derived in equation 4. Let the set of cluster centroids be denoted by $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K\}$, where K is the number of clusters. Each token's centroid is computed by minimizing the cosine distance between the token representations and the centroids.

3.3 Graph Construction

We construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ represents a cluster (or node) and the edges capture sequential relationships between tokens. Importantly, this graph is constructed based **only on the tokens in the NuminaMath dataset**. Specifically, given a sequence of cluster assignments $\{c_1, c_2, \dots, c_N\}$ corresponding to the token sequence, the weight of the edge between clusters i and j is defined as:

$$w_{i,j} = \sum_{t=1}^{N-1} \mathbb{I}\{c_t = i \wedge c_{t+1} = j\}, \tag{5}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function that evaluates to 1 if its argument is true and 0 otherwise.

The resulting graph enables us to analyze the structural relationships between clusters via the edge weights. Since we build this graph only on the NuminaMath dataset, this analysis can provide insights into critical patterns and features that are relevant to reasoning and math-related tasks.

We show an example of graph construction in figure 2.

3.4 Reward Model and the Explore-Exploit Trade-Off

We use a simple reward model defined as follows to show that our clustering and graph construction algorithm is grounded. Let the reward R for a prompt p be

$$R(p) = \sum_{i=0}^{N-1} w_{p_i, p_{i+1}} \tag{6}$$

where N is the number of tokens in prompt p and p_i is the i^{th} token in p. The weight $w_{i,j}$ is computed as in equation 5. This reward function quantifies the extent to which a prompt adheres to the sequential patterns prevalent in the NuminaMath dataset.

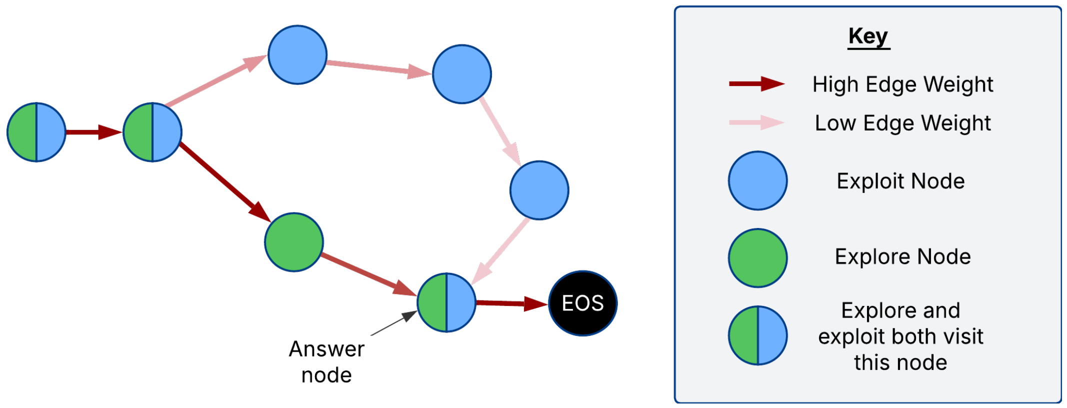


Figure 3: Graph showing example exploit vs. explore reasoning trajectories. Exploit takes the greediest approach, only transitioning through high weight edges. Explore takes lower weight edges but arrives at the same correct answer.

Justification of the Reward Model. The reward model is justified by its interpretability and its direct connection to the structural relationships between token clusters. Since each edge weight $w_{i,j}$ captures the frequency of transitions between clusters i and j, the cumulative reward $R(p)$ serves as an interpretable metric that reflects how well a given prompt aligns with established reasoning pathways; certain sequences of reasoning steps are more likely to lead to correct solutions.

Explore vs. Exploit in the Graph. In complex reasoning tasks, balancing exploration and exploitation is critical. In our framework, exploitation corresponds to following the highest weighted edges in the graph, which represent well-established and frequently observed token transitions. This type of greedy strategy reinforces conventional reasoning patterns that have proven successful in the NuminaMath dataset.

Conversely, exploration involves deliberately probing transitions that are less frequent. Although these transitions may not be optimal in a strictly greedy sense, they can lead to the discovery of alternative reasoning pathways that might yield improved or more creative solutions. We illustrate this trade-off in figure 3.

3.5 Metrics

Let us define G as the set of generated token sequences and O as the set of original token sequences.

To assess the quality of the G against O, we employ a combination of metrics that evaluate fidelity, diversity, and structural similarity. These metrics are designed to provide a comprehensive understanding of how well the generated sequences align with the original data in terms of content, variability, and structural patterns.

3.5.1 Diversity Metrics

Entropy. Entropy is used to measure the diversity of cluster frequencies in the generated sequences. Higher entropy indicates greater diversity, which is desirable as it suggests that the generated sequences are not overly repetitive and exhibit a variability similar to that of the original sequences. The entropy of the generated sequences is compared to that of the original sequences to check the model's diversity in generation.

3.5.2 Alignment Metrics

Dynamic Time Warping (DTW). DTW is employed to measure the alignment between sequences in G and O. For each generated sequence, the minimum DTW distance from the original sequences is calculated using the SAE representation. The average of these distances provides a quantitative measure of structural similarity. Lower DTW distances indicate better alignment, suggesting that the generated sequences closely follow the structural patterns of the original data.

Distance Distribution. The KL divergence is also used to compare distance distributions between consecutive tokens in the sequences. This metric evaluates whether the generated sequences maintain the same relational patterns between tokens as observed in the original sequences. Two approaches are used for this comparison: one using the direct representation of each element (SAE) and another using centroid-based representations obtained through k-means clustering.

4 Results & Analysis

4.1 Graph-based Validation Metrics for Exploitation

We compute the average reward per token for different datasets using the reward formulation described in equation 5) to evaluate how well our graph-based reward model captures established reasoning pathways. Table 1 summarizes the results for three models: MiniCPM-2B-128k, MiniCPM-2B-sft-bf16 and MiniCPM-1B-sft-bf16 (Hu et al., 2024).

Dataset	MiniCPM-2B-128k	MiniCPM-2B-sft	MiniCPM-1B-sft
GSM8K	189.40	74.02	124.28
HumanEval	120.81	43.47	84.27
Wikipedia	93.32	5.27	11.43
NuminaMath	415.97	186.74	424.75

Table 1: Average per-token rewards computed using our graph-based reward model over 100 prompts per dataset. Higher rewards on reasoning-centric datasets (e.g., NuminaMath, GSM8K, HumanEval) versus lower rewards on non-reasoning data (Wikipedia) indicate stronger alignment with established reasoning patterns for each model.

Our results show that both models easily achieve the highest average rewards on the NuminaMath dataset; this should be given since we built our graph on this dataset.

However, we also see that reasoning patterns carry over to datasets like GSM8K (Cobbe et al., 2021) and HumanEval (Chen et al., 2021), with relatively high scores. Their contrast to the significantly lower rewards for the Wikipedia dataset (83.32, 5.27, 11.43), especially on the SFT models, demonstrates that the reward model is still sensitive to reasoning patterns across different types of data. Tokens that do not follow the prevalent reasoning pathways receive much lower rewards.

These results confirm that exploitation, i.e., greedily following the high-reward transitions encoded in the graph, can be a viable approach for guiding LLM generation in math and reasoning-related tasks. An exploitation-based approach may lead to more coherent and effective outputs. Overall, these metrics ground our graph construction method and show that higher rewards can help guide better generation.

4.2 Comparative Analysis

We evaluate the three MiniCPM language models on their ability to generate sequences that match the structural and distributional properties of the original data. The evaluation employs multiple metrics to assess accuracy, structural alignment (via DTW), distributional similarity (via KL divergence), and diversity (via entropy).

4.2.1 Model Performance Overview

The supervised fine-tuned models (MiniCPM-1B-sft and MiniCPM-2B-sft) demonstrate superior accuracy compared to the context-extended model (MiniCPM-2B-128k), as shown in Table 2. This comparison suggests that supervised fine-tuning has a more significant positive impact on sequence generation quality than extending the context length.

4.2.2 Structural Alignment Analysis

Dynamic Time Warping (DTW) distances reveal distinct patterns across models. The MiniCPM-1B-sft model shows the lowest DTW distances, with correct sequences averaging 4.66 and incorrect

Model	Accuracy (%)	Avg. Correct DTW	Avg. Incorrect DTW
MiniCPM-2B-128k	39.87	17.05	20.23
MiniCPM-1B-sft	52.00	4.66	5.55
MiniCPM-2B-sft	53.40	31.35	36.87

Table 2: Comparison of model performance across sequence generation tasks, showing accuracy (in %) and Dynamic Time Warping (DTW) distances for both correct and incorrect sequences. Lower DTW distances indicate better structural alignment with original sequences.

sequences averaging 5.55, indicating superior structural alignment with the original sequences. In contrast, MiniCPM-2B-sft exhibits the highest DTW distances (31.35 for correct sequences, 36.87 for incorrect sequences), despite having better accuracy. The MiniCPM-2B-128k model demonstrates intermediate DTW distances. Across all models, correct sequences consistently show lower DTW distances than incorrect ones, validating DTW's effectiveness as a quality metric.

4.2.3 Distributional Similarity

Model	SAE-based KL Divergence			Centroid-based KL Divergence		
	Corr-Orig	Incorr-Orig	Corr-Incorr	Corr-Orig	Incorr-Orig	Corr-Incorr
MiniCPM-2B-128k	0.178	0.391	0.236	0.086	0.258	0.235
MiniCPM-1B-sft	0.033	0.148	0.054	0.296	0.387	0.155
MiniCPM-2B-sft	0.068	0.157	0.011	0.357	0.375	0.300

Table 3: KL divergence measurements comparing distributional similarities between generated and original sequences. Measurements are shown for both direct representation (SAE) and centroid-based approaches. Lower values indicate better preservation of original sequence patterns. Corr-Orig: Correct vs Original, Incorr-Orig: Incorrect vs Original, Corr-Incorr: Correct vs Incorrect sequences.

KL divergence measurements were conducted using both direct representations (SAE) and clustered centroids, as detailed in Table 3. SAE-based analysis shows the lowest KL divergence between correct and original sequences to be from the MiniCPM-1B-sft model (0.033), indicating superior preservation of the original distance distribution. MiniCPM-2B-128k exhibits the highest divergence (0.178), while MiniCPM-2B-sft maintains moderate divergence (0.068). Interestingly, the centroid-based analysis reveals different patterns, with MiniCPM-2B-128k showing the lowest KL divergence between correct and original sequences (0.086), while both fine-tuned models show higher centroid-based divergences. The distributional patterns (Figures 4 and 5) demonstrate that incorrect generations tend to exhibit higher peak densities and narrower distributions, suggesting more repetitive patterns compared to both correct generations and original sequences.

4.2.4 Entropy Analysis

Model	Original	Correct	Incorrect
MiniCPM-2B-128k	5.50	5.68	5.87
MiniCPM-1B-sft	5.41	5.42	5.58
MiniCPM-2B-sft	5.62	5.51	5.64

Table 4: Entropy measurements quantifying sequence diversity across different models. Values are shown for original sequences and both correct and incorrect generations. Values closer to the original indicate better preservation of original sequence diversity patterns.

The entropy measurements assess sequence diversity, as presented in Table 4. The MiniCPM-1B-sft model demonstrates the closest entropy match with the original sequences, while MiniCPM-2B-128k shows slightly higher entropy than the original, indicating potential over-diversification. The MiniCPM-2B-sft model shows balanced entropy levels close to the original distribution.

4.3 Sequence Length Analysis

The analysis of sequence lengths reveals important differences between correct and incorrect generations across models, as shown in Table 5. The original sequences have an average length of 130.56 tokens, providing a baseline for comparison. The MiniCPM-1B-sft model demonstrates

Model	Original Length	Correct Length	Incorrect Length
MiniCPM-2B-128k	130.56	148.08	206.85
MiniCPM-1B-sft	130.56	132.77	177.83
MiniCPM-1B-sft	130.56	125.52	149.39

Table 5: Comparison of average sequence lengths across different models and generation categories. The original sequence length serves as a reference point for evaluating generation fidelity. Notably, incorrect generations consistently produce longer sequences across all models, suggesting a correlation between sequence length deviation and generation quality.

the closest length alignment for correct generations (132.77 tokens), with a 1.7% deviation from the original length. The MiniCPM-2B-sft model slightly underestimates sequence lengths (125.52 tokens), while 2B-128k shows significant overestimation (148.08 tokens). Notably, incorrect generations consistently produce longer sequences across all models, with the most pronounced effect in MiniCPM-2B-128k (206.85 tokens, 58.4% longer than the original). This pattern suggests that sequence length deviation might serve as an additional indicator of generation quality.

5 Conclusion

From our experiments, MiniCPM-2B-sft achieves the highest accuracy: it shows larger structural deviations (DTW distances), suggesting a potential trade-off between accuracy and structural preservation. This is most likely because adherence to the reference graph and creative reasoning traces that lead to high-quality generation and the correct answers couldn't coexist. The divergence between SAE and centroid-based measurements also highlights the importance of considering multiple representation levels (token-level vs. cluster-based) when evaluating sequence generation quality.

These results show that exploitation-based methods alone are insufficient to guide generation quality. Instead, balancing both exploitation and exploration is crucial to guide model generation to search for the correct answer. Our proposed SAE-based technique is a scalable way to supervise intermediate token-level generation and balance exploitation and exploration through reward models.

Looking forward, there are many possible improvements to this framework that future research can resolve. The most immediate improvement is likely the design of a better reward function. We imagine that metrics that combine rewards calculated from the graph, cosine distance between consecutive tokens, and other multi-level structural and semantic evaluations can help us better understand and refine our proposed tradeoff between exploitation and exploration.

Another promising direction is to develop integrated metrics combining structural, distributional, and semantic aspects to provide a more comprehensive quality assessment. Particularly, investigating the relationship between distributional patterns and generation quality could lead to better early detection mechanisms for incorrect generations.

Given the consistent length and distributional deviations in incorrect generations, we can also look towards developing techniques to explicitly diverge from the ”incorrect” pathway during generation in order to help improve output quality. The clear differences in distributional patterns between correct and incorrect generations also suggest the potential to develop better, more interpretable indicators.

Overall, we hope that these insights using our SAE-based clustering and graph-based technique can help lead the way toward more efficient and higher quality reasoning systems.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL `https://aclanthology.org/2024.acl-long.662/`.

Kimi AI. Kimi k1.5: Scaling reinforcement learning with large language models. *arXiv preprint arXiv:2501.12599*, 2025. URL `https://arxiv.org/abs/2501.12599`.

Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. Numinamath 7b tir. `https://huggingface.co/AI-MO/NuminaMath-7B-TIR`, 2024.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*, 2023.

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.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

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. URL `https://arxiv.org/abs/2406.04093`.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL `https://arxiv.org/abs/2501.12948`.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL `https://arxiv.org/abs/2404.06395`.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Jinyi Liu, Yi Ma, Jianye Hao, Yujing Hu, Yan Zheng, Tangjie Lv, and Changjie Fan. Prioritized trajectory replay: A replay memory for data-driven reinforcement learning. *arXiv preprint arXiv:2306.15503*, 2023. URL `https://arxiv.org/abs/2306.15503`.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, and Pengfei Liu. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023. URL `https://arxiv.org/abs/2310.10080`.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Luke Marks, Alisdair Paren, David Krueger, and Fazl Barez. Enhancing neural network interpretability with feature-aligned sparse autoencoders. *arXiv preprint arXiv:2411.01220*, 2024.

OpenAI. Learning to reason with llms. OpenAI Blog, 2024a. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. Advancing reflective reasoning in language models. *OpenAI Blog*, 2024b. URL https://openai.com/blog/advancing-reflective-reasoning-in-language-models/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. URL https://arxiv.org/abs/2408.03314.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. 2025.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv preprint arXiv:2308.11761*, 2023.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

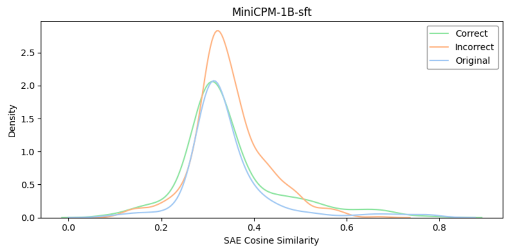
Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=j7DZWSc8qu.

Jingyuan Yang, Rongjun Li, Weixuan Wang, Ziyu Zhou, Zhiyong Feng, and Wei Peng. Lf-steering: Latent feature activation steering for enhancing semantic consistency in large language models. *arXiv preprint arXiv:2501.11036*, 2025.

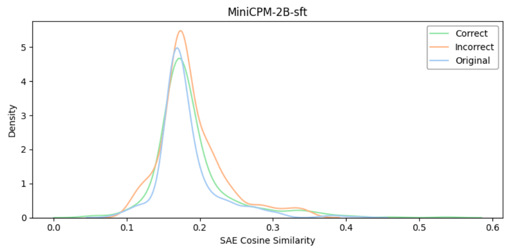
Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Qingyu Yin, Chak Tou Leong, Hongbo Zhang, Minjun Zhu, Hanqi Yan, Qiang Zhang, Yulan He, Wenjie Li, Jun Wang, Yue Zhang, et al. Direct preference optimization using sparse feature-level constraints. *arXiv preprint arXiv:2411.07618*, 2024.

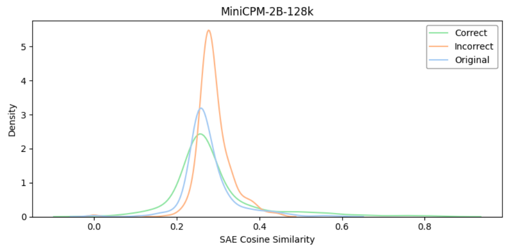
A Appendix



(a) MiniCPM-1B-sft

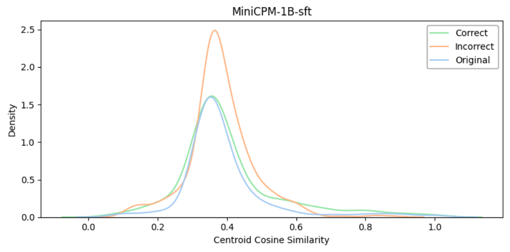


(b) MiniCPM-2B-sft

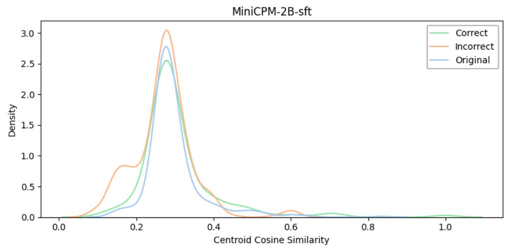


(c) MiniCPM-2B-128k

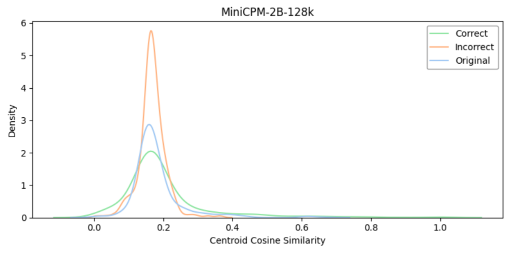
Figure 4: Distribution of SAE cosine similarities between consecutive sequence elements across different models. Each subplot compares distributions between correctly generated sequences (green), incorrectly generated sequences (orange), and original sequences (blue). (a) MiniCPM-1B-sft shows relatively lower peak density but better alignment between correct and original distribution, (b) MiniCPM-2B-sft demonstrates higher peak density with better distributional alignment, and (c) MiniCPM-2B-128k exhibits the highest peak density but shows notable deviation from the original distribution pattern. The curves represent kernel density estimations, revealing the underlying probability distribution of the cosine similarities. Higher similarity values indicate stronger semantic relationships between consecutive elements in the sequences.



(a) MiniCPM-1B-sft



(b) MiniCPM-2B-sft



(c) MiniCPM-2B-128k

Figure 5: Distribution of centroid cosine similarities between consecutive sequence elements across different models. Each subplot compares distributions between correctly generated sequences (green), incorrectly generated sequences (orange), and original sequences (blue). (a) MiniCPM-1B-sft shows relatively lower peak density but better alignment between correct and original distribution, (b) MiniCPM-2B-sft demonstrates higher peak density with better distributional alignment, and (c) MiniCPM-2B-128k exhibits the highest peak density but shows notable deviation from the original distribution pattern. This centroid-based analysis complements the fine-grained SAE similarity distributions shown in Figure 4.