Internal states before *wait* modulate reasoning patterns

Anonymous Authors[1]

Abstract

Prior work has shown that a significant driver of performance in reasoning models is their ability to reason and self-correct. A distinctive marker in these reasoning traces is the token *wait* , which often signals reasoning behavior such as backtracking. Despite being such a complex behavior, little is understood of exactly why models do or do not decide to reason in this particular manner, which limits our understanding of what makes a reasoning model so effective. In this work, we address the question whether model's latents preceding *wait* tokens contain relevant information for modulating the subsequent reasoning process. To this end we train crosscoders at multiple layers layers of `DeepSeek-R1-Distill-Llama-8B` and its base version, and, introduce a novel latent attribution patching technique for the crosscoder setting. Using our technique, we locate a small set of features relevant for promoting/surpressing *wait* tokens' probabilities. Finally, through a targeted series of experiments analyzing max-activating examples and causal interventions, we show that many of our identified features indeed are relevant for the reasoning process and give rise to different types of reasoning patterns such as *restarting from the beginning, recalling prior knowledge, expressing uncertainty, and double-checking*.

1. Introduction

A growing class of language models known as reasoning models, for instance, DeepSeek-R1 (DeepSeek-AI et al., 2025), OpenAI o1 series (OpenAI et al., 2024), and others (Team, 2024; Kavukcuoglu, 2025; Anthropic, 2025), produce detailed internal reasoning chains before generating responses. While they showcase sophisticated reasoning capabilities, our understanding of their internal reasoning mechanisms remains limited. A recent work by Venhoff et al. (2025) categorized DeepSeek R1's reasoning process into behavioral patterns such as example testing, uncertainty estimation, and backtracking to show how models solve reasoning tasks. This leads us to ask the following question: when and how do these reasoning patterns form?

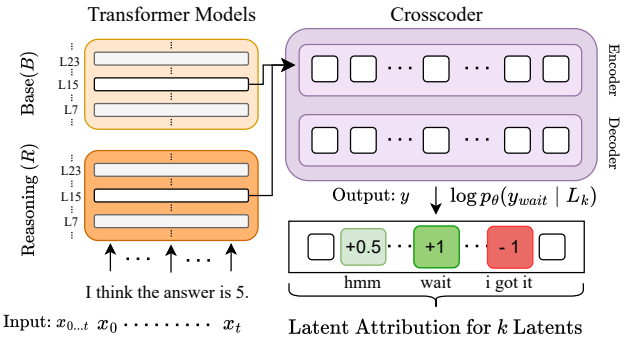

Figure 1. **We determine reasoning related crosscoder features** via latent attribution with respect to *wait* tokens' logits.

This is a broad question that encompasses both the training dynamics, such as when and how reasoning circuits form, and the emergence of reasoning patterns and behaviors during inference. In this work, we focus on the latter, specifically within the distilled R1 model `DeepSeek-R1-Distill-Llama-8B`.[1] From observation, in this model the token *wait* is strongly associated with the self-reflection of the model (Baek & Tegmark, 2025), which could relate to reasoning patterns such as backtracking, deduction, and uncertainty estimation (Venhoff et al., 2025). Hence, we hypothesize that the features that modulate this token prediction can be useful in understanding the shifts in the reasoning patterns of the reasoning model.

Contributions. Motivated by this observation, we train Sparse Crosscoders (Lindsey et al., 2024) a recent mechanistic interpretability technique that allows to learn features within a paired base and finetuned model in an unsupervised way and can be used to perform model diffing.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]Although we work with the distilled model, we will refer to it as R1 for succinctness.

We leverage our crosscoders to discover latent features that modulate reasoning behaviors. To reduce the search space, we focus on features promoting and suppressing *wait* tokens that we hypothesize to play a key role in R1's reasoning. To this end, we extend Attribution Patching (Kramár et al., 2024) to the crosscoder setting. Our crosscoder latent attribution method allows us to efficiently score all of our 32,768 features regarding their contribution to the logits of *wait* tokens. As a result, we obtain two short lists of relevant features, the 50 top features with the largest contributions to *wait* logits and the 50 bottom ones, that we can effectively investigate.

By utilizing the crosscoders classification of the features in to `base, shared`, and `finetuned/reasoning` we found that surprisingly many reasoning related features fall inside of the bottom bucket, suggesting that both amplifying as well as suppressing *wait* are of key importance to reasoning. By studying max-activating examples we observe a similar pattern. While top features mostly occur in the frequently reported backtracking and self-verification behaviors, we again notice that bottom features stand out in terms of the diversity of the reasoning behaviors within which they occur (e.g., starting again from an earlier step, expressing uncertainty, wrapping up, recalling knowledge, etc.). Finally, we perform a causal analysis via activation steering, in which we observe effects matching our interpretations based on the max-activating examples for many features, in particular the ones mentioned in parenthesis before. Our work showcases how to reduce the combinatorial complexities of crosscoder feature based interpretability via the help of our novel latent feature attribution technique and thereby opens the door for future researchers to explore finetuned models such as the nowadays prevalent chat and reasoning models.

2. Methods

To discover features, we create reasoning data instances (Section 2.2) and train Sparse Crosscoders (Section 2.1). We then apply Latent Attribution patching (Section 2.3) to filter features that strongly modulate *wait* tokens. Finally, we steer these features (Section 2.4) to evaluate their impact on the model's reasoning behavior, helping us determine how actionable they are.

2.1. Sparse Crosscoder

Sparse Autoencoder (SAE) decomposes model activations into a sparse set of linear feature directions. Sparse Crosscoder extends this by decomposing activations from different layers, models, or context positions. To explore feature correspondence between two models, we train L1 Sparse Crosscoders (Lindsey et al., 2024) between `DeepSeek-R1-Distill-Llama-8B` and its base version. We train three Crosscoders at 25%, 50%, and 75% layers depth, respectively, using an expansion factor of 8 to learn 32768 features. For training details, refer to Appendix A.

2.2. Reasoning Instances with *wait*

Venhoff et al. (2025) released rollouts on reasoning problems from `DeepSeek-R1-Distill-Llama-8B` that have reasoning traces and 193 out of 500 samples have either of the following *wait* tokens — " Wait", "Wait", " wait", "wait". For every *wait* in every sample that has a preceding reasoning sequence, we create a subsequence from the beginning of the sentence to the token right before *wait*. This filters for *wait* tokens related to reasoning behavior. This generates 350 subsequences.

2.3. Latent Attribution Patching

As shown in Figure 1, latent attribution patching observes how much each latent component in the crosscoder contributes to the change in the downstream metric,M_{patch}, which we define as the log probability of the sum of the four tokens of *wait* as follows:

$$M_{patch} = \log p_\theta(y_{wait} \mid L) \tag{1}$$

where $L \in \mathbb{R}^{batch \times seq \times dim}$ are the latents obtained from the crosscoder, and y_{wait} denotes the token sequence corresponding to *wait*, which includes " Wait", "Wait", " wait", "wait". See more info on the setup at Appendix B.

2.4. Steering

After training the columns of the decoder matrix $W^{(R)} \in \mathbb{R}^{d \times n_f}$ can be thought of as n_f linear feature directions/features $W^{(R)} = (\mathbf{f}_1, \ldots, \mathbf{f}_{n_f})$. These features can be used to modulate the model's rollouts by adding them to the model's activations during the generation of new tokens:

$$a^{(R)}_{\text{steered}}(x)_i = \begin{cases} a^{(R)}(x)_i + \alpha \frac{\|a^{(R)}(x)\|}{\|\mathbf{f}_k\|}\mathbf{f}_k, & \text{if i} \geq \text{t} \\ a^{(R)}(x)_i, & \text{otherwise,} \end{cases} \tag{2}$$

in which we "steered" with strength $\alpha \in \mathbb{R}$ and the kth feature at the last token of the input $x_1, \ldots, x_t$ and each newly generated token position $x_{t+1}, \ldots$.

3. Results

Intermediate decoding. As a first check we investigate our selected features through the patchscope-lens (Ghandeharioun et al., 2024). The patchscope-lens is an intermediate decoding technique that decodes (some) of the information contained in a latent by inserting it at its corresponding layer in a parallel forward pass that is processing a patchscope-lens prompt. Since we are using a reasoning model, we

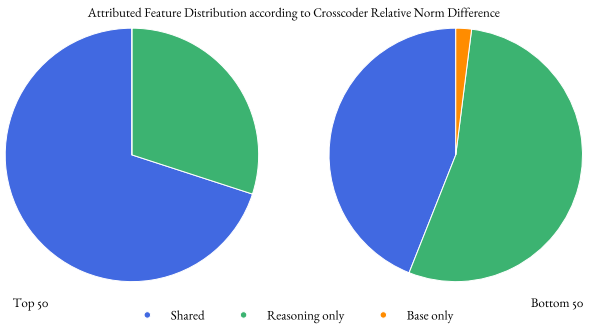

Figure 2. **Crosscoder classification of the features into `base`, `shared`, `finetuned`.**

slightly adapt the patchscope-lens prompt to the reasoning model's format and thought prefilling (see exact prompt in Appendix C). For each of the top/bottom features we compute the patchscope's next token distribution, then we take the average within the top/bottom groups. As a result, we obtain Fig. 5 (see Appendix). As can be seen top features indeed promote "Wait" and related tokens, whereas, bottom ones share " " as top token.

Model diffing. Crosscoders by design allow to classify their learned features into `base`, `shared`, and `finetuned`. As Minder et al. (2025) have recently shown this classification is not perfect, we ask the reader to take Fig. 2 in the appendix with a grain of salt. Fig. 2 shows the fraction of top and bottom features classified into the three categories. Both top and bottom features contain a significant number of `shared` features, which is expected, since most of the features learned by the crosscoder are in this category. The bottom features also contain some `base`-only features, - we hypothesize that some of the features that only the base model uses would decrease the likelihood of the *wait* token in reasoning sequences, since the base model was not trained to predict the *wait* token in such context. As expected, none of the top features are `base`-only. Most interestingly, the bottom features contain a larger number for `finetuned`-only, i.e., `reasoning` features. This suggests reasoning tuning allocates a substantial number of features to both suppressing as well as promoting *wait* token's probabilities.

Max activating examples. Next, we examine the features' max activating examples, which we obtain by computing feature coefficients over a dataset of 20 million tokens. We manually expect a set of 100 max activating examples and generate an automated annotation for each of our 100 features. We observe that the Top features largely contribute to backtracking (max activating examples activate on *wait* and "But" tokens). Bottom features correspond to behavior which can be interpreted as the model restarting its thinking process or concluding the reasoning trace. For example, feature #1565 activates on full stops at the end of the reasoning sequence and feature #32252 activates on a final answer at the end of samples with mathematical reasoning.

Feature steering. Finally, we investigate features' causal impact on the rollouts in the reasoning model by performing steering. We steer on a single input example, see Fig. 3. To match the setting in which we extracted the features we start steering the rollout from the token before the first *wait* token and from there generate up to 200 tokens while steering. Since steering is sensitive to the steering strength hyperparameter, we steer with multiple strengths -1.5, -1.25, -1.0, -0.75, -0.5, 0.5, 0.75, 1, 1.25, 1.5 for each feature. This results in 10 different continuations for each of the 100 features. In Fig. 4 we verified that the causal role of our features is broadly aligned with our expectations about them based on our feature selection criterion. In particular, we measure the number of characters occurring before the first *wait* . Since we selected our features based on latent attribution with respect to *wait* for top features the number of characters should be small when steering positively and for bottom features it should be large. Similarly, for features that are meaningful into both directions, which does not necessarily have to be the case because in (A) they feature coefficients are always positive, steering with negative strength should lead to larger distance to *wait* for top features and smaller one for bottom features. Fig. 4 matches these expectations.

Steered generations. We show example continuations under our interventions in Fig. 3. As can be seen, in particular, the bottom features lead to interesting reasoning patterns that we have not seen before in the literature. Among the top features we observed many instances of steering positively quickly resulting in degenerate sequences like "WaitWaitWait..." or " wait wait wait...". Steering these into the negative direction leads to *wait* disappearing from the outputs. Our best guess is that those features literally contribute to the *wait* tokens and it is unclear whether they can also trigger any specific reasoning patterns that would be comparable to the ones reported in Fig. 3. Additionally, we found several features that when steered positively/negatively lead towards the model wrapping up and providing the final response and when steered into the opposite direction lead to extended reasoning.

4. Related Work

With respect to understanding reasoning models, Venhoff et al. (2025) looks into the chain-of-thought texts and categorizes the trace into various reasoning behavior patterns such as deduction, adding knowledge, and example testing. They further find steering vectors to increase or decrease the appearance of a particular reasoning behavior. Baek & Tegmark (2025) investigates reasoning features, learned by crosscoders, via qualitative analysis of max activating

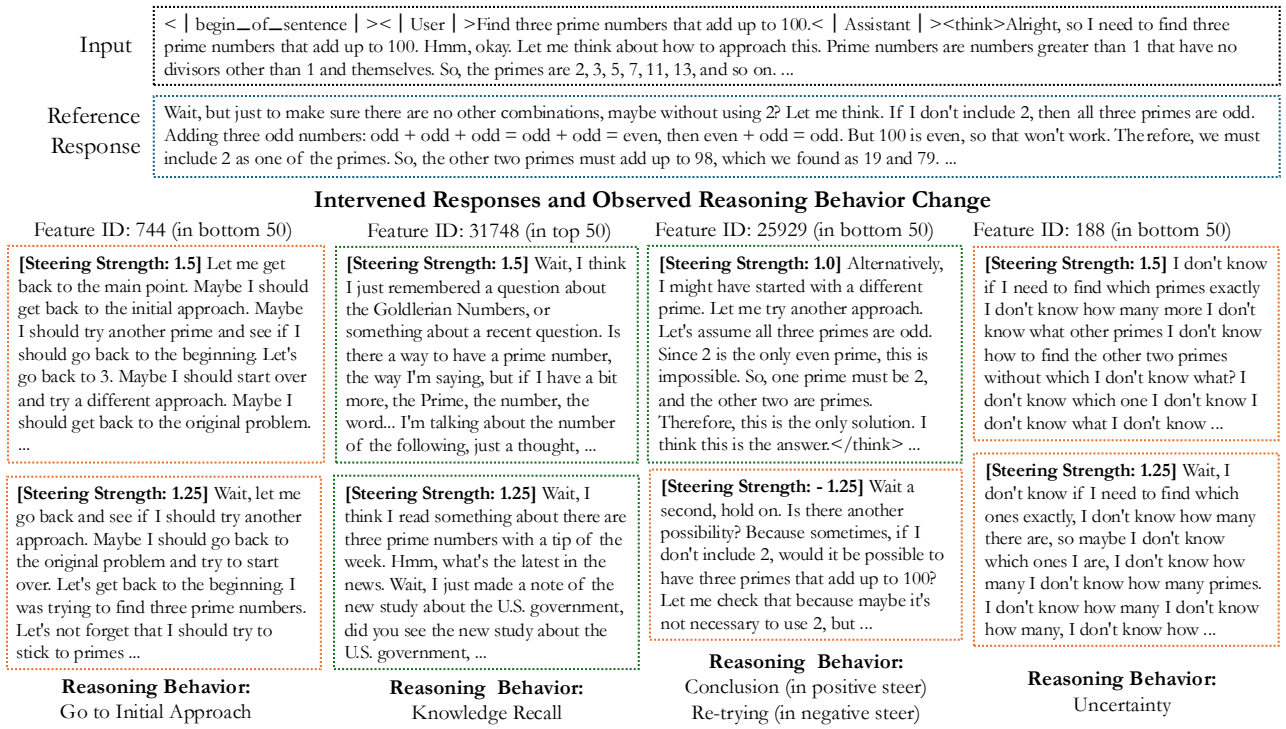

Figure 3. Change in reasoning behavior observed when certain features are steered in the positive and/or negative directions.

Figure 4. We compute how many characters occur before the first *wait* token in the continued rollout, while steering with all of our features and different intervention strengths. Steering with positive coefficient for the top features slightly increases the distance to the first *wait* which is due to oversteering, for bottom features as one would expect the distance to the first *wait* increases significantly. Negative steering has the opposite effect.

examples. An additional perspective on token-level and feature-level analysis of reasoning features is a concurrent work by Lee et al. (2025) that performs a case study on the self-verification process of a task-specific reasoning model. Similarly, Zhang et al. (2025) show that reasoning models "know" when they are right by training linear probes that can predict the correctness of intermediate solutions from the models' internals with high accuracy. In contrast to (Venhoff et al., 2025; Lee et al., 2025; Zhang et al., 2025) that focus on specific features and reasoning behaviors on labeled datasets, we are using crosscoders to discover features relevant to distilled reasoning model's "thought" process with minimal supervision. Baek & Tegmark (2025) also use crosscoders, however, in this work we focus on a more narrow set of the features that we obtain by our latent attribution technique that selects features most relevant for modulating the logits of different "wait" tokens.

5. Conclusion

Our study sheds light on how the model's latents preceding *wait* tokens signal behaviors like backtracking. We introduce latent attribution patching for identifying and testing which internal features influence these tokens in crosscoder models. We show that these features not only predict *wait* tokens but also shape how the model reasons.

Impact Statement

This work advances our understanding of how language models perform reasoning by uncovering internal features that influence key reasoning behaviors such as uncertainty and backtracking. By sharing a method to identify and manipulate these features, we provide tools for improving interpretability and guiding model behavior. This has broad implications for applications that require trustworthy AI, such as education and scientific discovery.

References

Anthropic. Claude 3.7 sonnet system card, February 2025. URL https://assets.anthropic.com/m/785e231869ea8b3b/original/claude-3-7-sonnet-system-card.pdf.

Baek, D. D. and Tegmark, M. Towards understanding distilled reasoning models: A representational approach. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=UYZCcnwgc4.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Ghandeharioun, A., Caciularu, A., Pearce, A., Dixon, L., and Geva, M. Patchscopes: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*, 2024.

Kavukcuoglu, K. Gemini 2.5: Our most intelligent ai model, 2025. URL https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

Kramár, J., Lieberum, T., Shah, R., and Nanda, N. Atp*: An efficient and scalable method for localizing llm behaviour to components, 2024. URL https://arxiv.org/abs/2403.00745.

Lee, A., Sun, L., Wendler, C., Viégas, F., and Wattenberg, M. The geometry of self-verification in a task-specific reasoning model, 2025. URL https://arxiv.org/abs/2504.14379.

Lindsey, J., Templeton, A., Marcus, J., Conerly, T., Batson, J., and Olah, C. Sparse crosscoders for cross-layer features and model diffing, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Minder, J., Dumas, C., Chughtai, B., and Nanda, N. Latent scaling robustly identifies chat-specific latents in crosscoders. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025. URL https://openreview.net/forum?id=JGRtSAlQ3h.

OpenAI, :, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., Iftimie, A., Karpenko, A., Passos, A. T., Neitz, A., Prokofiev, A., Wei, A., Tam, A., Bennett, A., Kumar, A., Saraiva, A., Vallone, A., Duberstein, A., Kondrich, A., Mishchenko, A., Applebaum, A., Jiang, A., Nair, A., Zoph, B., Ghorbani, B., Rossen, B., Sokolowsky, B., Barak, B., McGrew, B., Minaiev, B., Hao, B., Baker, B., Houghton, B., McKinzie, B., Eastman, B., Lugaresi, C., Bassin, C., Hudson, C., Li, C. M., de Bourcy, C., Voss, C., Shen, C., Zhang, C., Koch, C., Orsinger, C., Hesse, C., Fischer, C., Chan, C., Roberts, D., Kappler, D., Levy, D., Selsam, D., Dohan, D., Farhi, D., Mely, D., Robinson, D., Tsipras, D., Li, D., Oprica, D., Freeman, E., Zhang, E., Wong, E., Proehl, E., Cheung, E., Mitchell, E., Wallace, E., Ritter, E., Mays, E., Wang, F., Such, F. P., Raso, F., Leoni, F., Tsimpourlas, F., Song, F., von Lohmann, F., Sulit, F., Salmon, G., Parascandolo, G., Chabot, G.,

Zhao, G., Brockman, G., Leclerc, G., Salman, H., Bao, H., Sheng, H., Andrin, H., Bagherinezhad, H., Ren, H., Lightman, H., Chung, H. W., Kivlichan, I., O'Connell, I., Osband, I., Gilaberte, I. C., Akkaya, I., Kostrikov, I., Sutskever, I., Kofman, I., Pachocki, J., Lennon, J., Wei, J., Harb, J., Twore, J., Feng, J., Yu, J., Weng, J., Tang, J., Yu, J., Candela, J. Q., Palermo, J., Parish, J., Heidecke, J., Hallman, J., Rizzo, J., Gordon, J., Uesato, J., Ward, J., Huizinga, J., Wang, J., Chen, K., Xiao, K., Singhal, K., Nguyen, K., Cobbe, K., Shi, K., Wood, K., Rimbach, K., Gu-Lemberg, K., Liu, K., Lu, K., Stone, K., Yu, K., Ahmad, L., Yang, L., Liu, L., Maksin, L., Ho, L., Fedus, L., Weng, L., Li, L., McCallum, L., Held, L., Kuhn, L., Kondraciuk, L., Kaiser, L., Metz, L., Boyd, M., Trebacz, M., Joglekar, M., Chen, M., Tintor, M., Meyer, M., Jones, M., Kaufer, M., Schwarzer, M., Shah, M., Yatbaz, M., Guan, M. Y., Xu, M., Yan, M., Glaese, M., Chen, M., Lampe, M., Malek, M., Wang, M., Fradin, M., McClay, M., Pavlov, M., Wang, M., Wang, M., Murati, M., Bavarian, M., Rohaninejad, M., McAleese, N., Chowdhury, N., Chowdhury, N., Ryder, N., Tezak, N., Brown, N., Nachum, O., Boiko, O., Murk, O., Watkins, O., Chao, P., Ashbourne, P., Izmailov, P., Zhokhov, P., Dias, R., Arora, R., Lin, R., Lopes, R. G., Gaon, R., Miyara, R., Leike, R., Hwang, R., Garg, R., Brown, R., James, R., Shu, R., Cheu, R., Greene, R., Jain, S., Altman, S., Toizer, S., Toyer, S., Miserendino, S., Agarwal, S., Hernandez, S., Baker, S., McKinney, S., Yan, S., Zhao, S., Hu, S., Santurkar, S., Chaudhuri, S. R., Zhang, S., Fu, S., Papay, S., Lin, S., Balaji, S., Sanjeev, S., Sidor, S., Broda, T., Clark, A., Wang, T., Gordon, T., Sanders, T., Patwardhan, T., Sottiaux, T., Degry, T., Dimson, T., Zheng, T., Garipov, T., Stasi, T., Bansal, T., Creech, T., Peterson, T., Eloundou, T., Qi, V., Kosaraju, V., Monaco, V., Pong, V., Fomenko, V., Zheng, W., Zhou, W., McCabe, W., Zaremba, W., Dubois, Y., Lu, Y., Chen, Y., Cha, Y., Bai, Y., He, Y., Zhang, Y., Wang, Y., Shao, Z., and Li, Z. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Team, Q. Qwq: Reflect deeply on the boundaries of the unknown, November 2024. URL https://qwenlm.github.io/blog/qwq-32b-preview/.

Venhoff, C., Arcuschin, I., Torr, P., Conmy, A., and Nanda, N. Understanding reasoning in thinking language models via steering vectors. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=OwhVWNOBcz.

Zhang, A., Chen, Y., Pan, J., Zhao, C., Panda, A., Li, J., and He, H. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025.

A. Crosscoder Training Details

To train a crosscoder using activations from two models, B and R, the crosscoder feature activation can be computed as follows:

$$f(x_t) = ReLU(\sum_{i=B,R} W_{enc}^{(i)} a^{(i)}(x_t) + b_{enc}^{(i)}) \quad (3)$$

$$a'^{(i)}(x_t) = W_{dec}^{(i)} f(x_t) + b_{dec}^{(i)} \quad (4)$$

where $W_{enc}^{(i)}$ is the model i's encoder matrix, $W_{dec}^{(i)}$ is model i's decoder matrix, $a^{(i)}$ is the model i's activation at token x_t and $a'^{(i)}(x)$ is the reconstructed activation.

The crosscoder training minimizes the loss that consists of the sum of reconstruction MSE and the sum of the per-feature decoder vector's L2 norm.

$$Loss = \sum_{i=B,R} ||a'^{(i)} - a^{(i)}||^2 + \sum_k f_k(x_t) \sum_{i=B,R} ||W_{dec,k}^{(i)}|| \quad (5)$$

B. Additional information on Latent Attribution Patching

The change in the metric is attributed to each latent in the crosscoder as:

$$\text{Attribution} = W_{dec}^R \cdot \frac{\partial M_{patch}}{\partial H} \odot L \quad (6)$$

where:

- W_{dec}^R are the decoder weights of the reasoning model
- $\frac{\partial M_{patch}}{\partial H}$ is the gradient of the metric with respect to the hidden state space, which is a stacked hidden state from both the base and the reasoning model.
- $\cdot$ is the batched projection, which is computed via Einstein summation.
- $\odot$ denotes the elementwise multiplication across the latent dimensions.

C. Patchscope-lens prompt

```
<begin of sentence><User>Continue the
following pattern: cat cat
1135 1135
hello hello
<Assistant><think>
Okay I need to complete: cat cat
1135 1135
hello hello
?
```

D. Additional Results

Checkout the additional figures for the results section here:

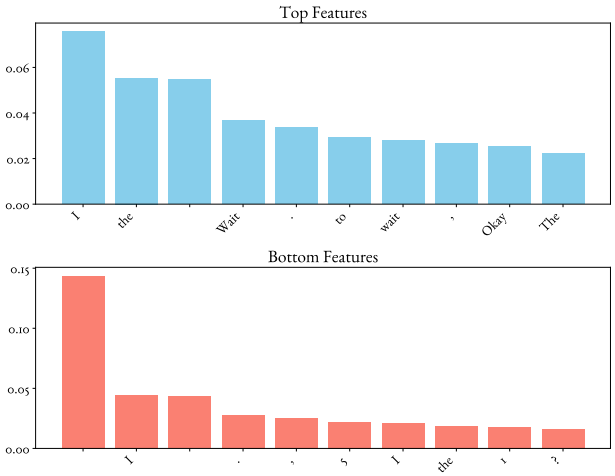

Figure 5. Top token probabilities as decoded by the patchscope. As can be seen **top features indeed promote "Wait" and related tokens**. In contrast **for bottom features no pattern is visible.**