
The Geometry of Self-Verification in a Task-Specific Reasoning Model

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 How do reasoning models verify their own answers? We study this question by
2 training a model using DeepSeek R1’s recipe on the Countdown task. We leverage
3 the fact that preference tuning leads to mode collapse, yielding a model that always
4 produces highly structured chain-of-thought sequences. With this setup, we do top-
5 down and bottom-up analyses to reverse-engineer how the model verifies its outputs.
6 Top-down, we find Gated Linear Unit (GLU) weights encoding verification-related
7 tokens, such as “success” or “incorrect”. Bottom-up, we find that “previous-token
8 heads” are mainly responsible for self-verification in our setup. Our analyses meet
9 in the middle: drawing inspiration from inter-layer communication channels, we
10 use the identified GLU weights to localize as few as six attention heads that can
11 disable self-verification, pointing to a necessary component of a potentially larger
12 verification circuit. Finally, we verify that similar verification components exist in
13 our base model and a general reasoning DeepSeek-R1 model.

14 1 Introduction

15 Recent language models demonstrate impressive reasoning capabilities. Such models are trained with
16 reinforcement learning (RL) in which they are rewarded when their final outputs are correct.

17 Behaviorally, these models generate long chain-of-thought (CoT) [37] reasoning traces. There is
18 an open question on whether monitoring their CoT is worthwhile, given a growing line of work
19 suggesting that their CoTs do not faithfully reflect the model’s inner computations [2, 14, 35]. Can we
20 monitor their hidden states instead? We take a step towards investigating this question, by studying a
21 model’s inner mechanism for a crucial reasoning step, i.e., self-verification.

22 General reasoning entails a broad range of tasks, requiring a diverse set of skills. In order to conduct
23 a systematic study, we train and analyze a task-specific reasoning model using the same recipes
24 from DeepSeek R1. We limit the scope of our study to a specific task that requires search – a core
25 reasoning skill broadly applicable for many reasoning tasks. We also select a task in which we can
26 expect the verification mechanism ahead of time, making our analyses easier. Namely, we study
27 Countdown [9, 10, 31, 38], in which a set of numbers (operands) and a target number is given, and
28 the model must find the right arithmetic combination using the operands to reach the target number.
29 Because the target number is specified in the context, we can expect attention heads to play a role in
30 verification and shed light onto other relevant weights and subspaces pertaining to self-verification.

31 Studying a task-specific model has a second non-obvious benefit: training language models with RL
32 (i.e., with preference signals) can lead to mode collapse towards majority preferences, significantly
33 reducing the diversity of their outputs [13, 21, 28, 34]. Luckily, in the context of model interpretability,
34 this means that our task-specific model converges to always generating well-structured CoT sequences,
35 allowing us to easily and systematically parse its reasoning trace (e.g., see Table 1).

36 We conduct “top-down” and “bottom-up” analyses to reverse-engineer how the model verifies its own
 37 outputs. Our two analyses meet in the middle, revealing key subspaces relevant for model verification.

38 Going top-down, we leverage linear probes to find Gated Linear Unit (GLU) vectors in late layers that
 39 often encode tokens relevant for verification. Interestingly, these vectors also seem to correlate with
 40 English or Chinese tokens, like “success” or “不完” (“failed”). Furthermore, the antipodal directions
 41 of these vectors also encode the antonyms of correct or incorrect tokens.

42 Going bottom-up, given the nature of our task, we hypothesize and verify that attention heads play a
 43 significant role. We find “previous-token heads” – attention heads that attend to previous occurrences
 44 of the current token – that attend to the provided solution in the context. Previous-token heads have
 45 been studied before, for instance in the context of induction heads [26]. Through causal analyses, we
 46 find that disabling previous-token heads disables model verification.

47 Our two analyses meet in the middle: we find that disabling previous-token heads also deactivates our
 48 GLU vectors. Inspired by inter-layer communication channels [7, 19], we look for previous-token
 49 heads that most align with the “receptive-field” of our GLU vectors, allowing us to localize as
 50 few as six attention heads that reliably disables model verification. Thus our work finds necessary
 51 components for a potentially larger verification circuit.

52 Finally, we verify that similar verification components exist in our base model prior to RL, as well as
 53 in a general reasoning model, DeepSeek-R1-Distill-Qwen-14B.

54 Obviously, most reasoning tasks do not provide an easily verifiable solution in the context. However,
 55 by illustrating a thorough mechanism of verification in our simplified setup, we take a step towards
 56 the possibility of monitoring and interpreting a model’s inner computations in its hidden states.

57 2 Notations, Key Terminologies

58 We first establish key terminologies and notations. A Transformer’s forward pass first embeds the
 59 input using weights $W_E \in \mathbb{R}^{d \times V}$. The embeddings go through L Transformer blocks, yielding
 60 hidden states $\mathbf{x}^\ell \in \mathbb{R}^d, \ell \in [L - 1]$. The last layer, $\mathbf{x}^{L-1}$, is “unembedded”, or projected back to
 61 the token embedding space using W_E , and the nearest neighboring token embedding of $W_E^\top \mathbf{x}^{L-1}$ is
 62 outputted. Each block consists of attention heads and Gated Linear Units (GLUs) [33].

63 **Attention.** Each attention head consists of key (W_K), query (W_Q), value (W_V), and output (W_O)
 64 weights. an attention pattern A is computed using key and query weights:

$$A = \text{softmax}(\mathbf{x}_i^\top W_Q^\top W_K \mathbf{x}_j) \quad (1)$$

65 where $W_Q^\top W_K$ is sometimes referred to as a “QK circuit”. A is used to scale the “OV circuit”
 66 ($W_O W_V$) to produce an output for each head:

$$h(\mathbf{x}) = (A \otimes W_O W_V) \cdot \mathbf{x} \quad (2)$$

67 **Gated Linear Units and GLU_{Out} Vectors.** Given a Gated Linear Unit (GLU) block:

$$\text{GLU}(\mathbf{x}) = (\phi(W_{gate}\mathbf{x}) \odot W_{up}\mathbf{x}) W_{out} \quad (3)$$

68 where ϕ is a non-linearity and $W_{gate}, W_{up}, W_{out} \in \mathbb{R}^{d_{glu} \times d}$, we decompose it as following:

$$M = \phi(W_{gate}\mathbf{x}) \odot W_{up}\mathbf{x}, M \in \mathbb{R}^{d_{glu}}, \quad (4)$$

$$\text{GLU}(\mathbf{x}) = M^\top W_{out} = \sum_j^{d_{glu}} m_j \mathbf{v}_j \quad (5)$$

70 where m_j is the j -th entry in M and $\mathbf{v}_j$ is the j -th row of W_{out} .

71 Importantly, the output of each GLU block can be thought of as the weighted sum of $\mathbf{v}$ ’s (the rows of
 72 W_{out}), where the weights are determined by activations M . **We refer to elements of M as GLU**
 73 **neurons and vectors $\mathbf{v}$ as GLU_{Out} vectors.** For more details, we refer readers to [7].

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process and then provides the user with the answer. User: Using the numbers [20, 14, 40], create an equation that equals 28. You can use basic arithmetic operations (+, -, *, /) and each number can only be used once. Show your work in `<think>` `</think>` tags. And return the final answer in `<answer>` `</answer>` tags, for example `<answer> (1 + 2) / 3 </answer>`.

Assistant: Let me solve this step by step. `<think>` We have the numbers 20, 14, and 40. We need to use these numbers to make an equation that equals 28 using basic arithmetic operations. Let's try different combinations:

40 - 14 - 20 = 10 - 20 = -10 (not 28)

40 - 14 + 20 = 26 + 20 = 46 (not 28)

40 + 14 - 20 = 54 - 20 = 34 (not 28)

40 + 14 + 20 = 54 + 20 = 74 (not 28)

40 * 14 / 20 = 560 / 20 = 28 (**this works**)

So, the equation that equals 28 is 40 * 14 / 20. `</think>` `<answer> (40 * 14) / 20 </answer>`

Table 1: **CountDown Task.** The model must find an arithmetic combination of the operands to reach the specified target number. We leverage the fact that preference tuning leads to mode collapse, resulting in a model that consistently generates structured CoT tokens that we can easily parse.

74 3 Training Task-Specific Reasoning Models

75 We apply DeepSeek R1-Zero’s setup with Qwen2.5-3B as our base model (Hyperparams: Appx. A).¹

76 Our task, CountDown, is a simple testbed frequently used to study recent reasoning models [9, 10, 31,
77 38] – given a set of 3 or 4 operands (e.g., 19, 36, 55, 7) and target number (e.g., 65), the task is to find
78 the right arithmetic combination of the operands to reach the target number (i.e., 55 + 36 - 7 - 19).

79 The model is given two rewards: accuracy reward for reaching the correct final answer, and a format
80 reward when it generates its CoT tokens in between “`<think>`” and “`</think>`” tokens. For more
81 details on how R1-Zero is trained, see [11]. We refer to our task-specific model as $R1_{\text{Down}}^{\text{Count}}$.

82 One advantage of studying a specific task is in that preference training leads to mode collapse [13,
83 21, 28, 34], resulting in a reduction in generation diversity. In our context, this is desirable, as the
84 model converges to generating a highly structured CoT sequence. See Table 1.

85 This allows us to easily parse the model’s CoT. Namely, the model enumerates through many attempts,
86 while always marking each attempt as either “(this works)” or “(not {ans})”. Thus, we can study the
87 model’s hidden states at specific timesteps, such as right before it produces either “this” or “not”,
88 which we refer to as t_{valid} and t_{invalid} . We refer to the hidden states at these timesteps as $\mathbf{x}_{\text{Valid}}$ and
89 $\mathbf{x}_{\text{Invalid}}$. We refer to the timestep in the prompt at which the target number is specified as t_{ans} .

90 4 Components for Self-Verification in CountDown

91 Here we present a series of analyses to identify weights and subspaces relevant for verification. We
92 do a “top-down” analysis to find relevant GLU vectors in late layers, and a “bottom-up” analysis to
93 find relevant attention heads in early layers. Our analyses meet in the middle, to identify relevant
94 subspaces for verification. We verify the role of such weights and subspaces via causal experiments.

95 4.1 Top-Down: Finding Verification-Related GLU Vectors

96 **LogitLens.** We start our analysis by applying LogitLens [23] to compare the hidden states of $\mathbf{x}_{\text{Valid}}$
97 and $\mathbf{x}_{\text{Invalid}}$ on a sample size of 300. We apply the unembedding layer at all intermediate layers $\mathbf{x}^{\ell}$
98 and inspect the resulting nearest neighboring tokens across 300 samples.

99 Figure 1(a, b) shows our results in the late layers (see Appendix Figure 5 for more layers). Interest-
100 ingly, we see tokens such as “SUCCESS”, “yes”, “bingo” show up for $\mathbf{x}_{\text{Valid}}$, and “不符合” (“Does
101 not conform”), “not”, “不合” (“Incompatible”) for $\mathbf{x}_{\text{Invalid}}$. Interestingly, we often observe English
102 tokens for $\mathbf{x}_{\text{Valid}}$ and Chinese tokens for $\mathbf{x}_{\text{Invalid}}$. What drives these tokens to appear?

¹We use TinyZero: <https://github.com/Jiayi-Pan/TinyZero/tree/main>

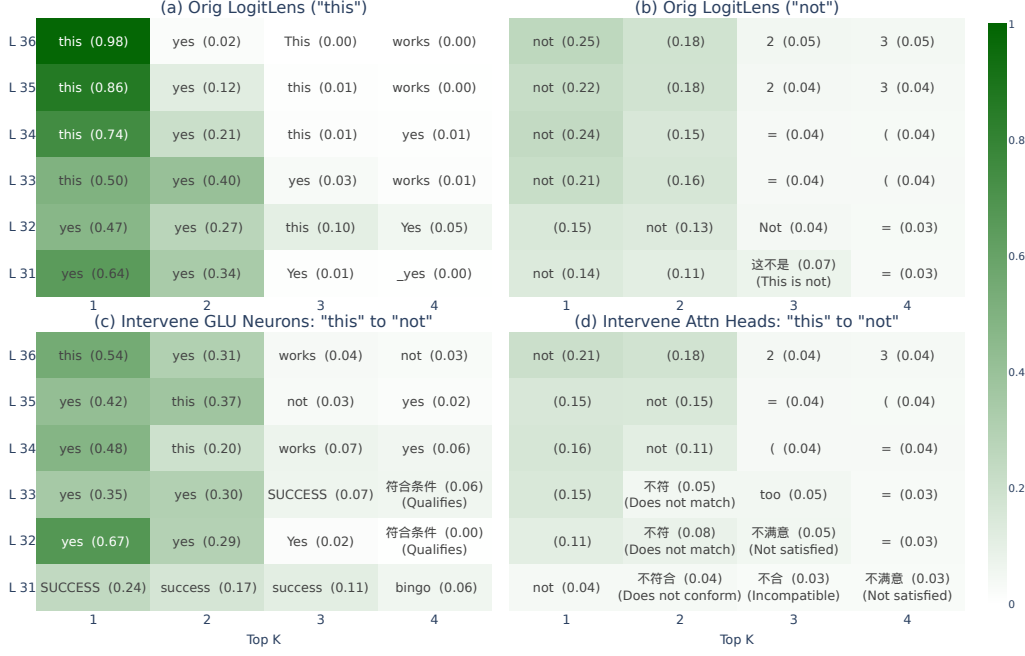


Figure 1: **Averaged LogitLens from 300 samples.** We see tokens related to verification (“success”, “不合”) in the last few layers. (a), (b) show the top tokens when (in)correct solutions are reached. (c), (d) shows results from intervening on either GLU weights or attention heads, given a correct solution. For (c), while the model is less certain (P(“this”) drops from 0.98 to 0.54), we still see tokens such as “success” showing up. For (d), we no longer see any tokens related to “success”, and the model’s final next-token predictions closely resembles when the model has not found a solution (b).

103 **Probing.** To answer this question, we train linear probes $W^\ell \in \mathbb{R}^{2 \times d}$ at every layer ℓ from
 104 timesteps right before “this” or “not” is predicted. These timesteps correspond to when an answer is
 105 produced, and an open parenthesis tokens “(” is being predicted next, as opposed to “this” or “not”.

106 W^ℓ is a linear mapping from the hidden states, $\mathbf{x}^\ell$, to a binary label of whether the model has found
 107 the solution. Our training data is $\mathcal{D} = \{(\mathbf{x}_{y^i}^\ell, y^i)\}_{i=0}^{N-1}$, $y^i \in \{0 \text{ (“not”)}, 1 \text{ (“this”)}\}$, $N=327,680$. We
 108 solve for W^ℓ to fit $y = \text{softmax}(W^\ell \mathbf{x}^\ell)$ using gradient descent (hyperparameters in Appendix C).

109 Validation accuracy ($N = 512$) per layer is provided in the Appendix (Figure 6), with accuracy
 110 usually staying above 90% after the first few layers. High accuracy suggests that our probing vectors
 111 $W[0]$, $W[1]$ identify a direction in the model’s activation space that linearly separates points of $\mathbf{x}_{\text{Valid}}$
 112 and points of $\mathbf{x}_{\text{Invalid}}$ (i.e., linearly separable subspaces).

113 Such vectors can steer the model. Simply adding $W[0]$ or $W[1]$ into hidden states can push $\mathbf{x}$ towards
 114 $\mathbf{x}_{\text{Valid}}$ or $\mathbf{x}_{\text{Invalid}}$, and change the model’s output to indicate that it has (or has not) found a solution,
 115 even when it has not (or has). We provide qualitative examples of steering results in Appendix E.

116 **GLU_{Valid}, GLU_{Invalid} Vectors.** Our probe W tells us that mid-layer activations can be linearly
 117 separated to identify solved cases ($\mathbf{x}_{\text{Valid}}$) from unsolved cases ($\mathbf{x}_{\text{Invalid}}$), but also serves a secondary
 118 purpose. Namely, we can use W to identify GLU_{Out} vectors of interest [15].

119 Per layer, we select the top $k (= 50)$ GLU_{Out} vectors by how similar they are to $W^\ell[0]$ or $W^\ell[1]$ using
 120 cosine similarity. One can consider these vectors as weights that contribute the most towards $W^\ell[0]$
 121 (no solution) or $W^\ell[1]$ (found solution) directions. We refer to them as GLU_{Invalid} and GLU_{Valid}
 122 vectors. This results in $k \times L \times 2$ GLU_{Valid, Invalid} vectors (0.9% of the model’s GLU_{Out} vectors).

123 Unembedding GLU_{Valid/Invalid} vectors reveal which tokens get promoted when they are activated.
 124 Table 2 shows their nearest neighbors in the model’s token embedding space. We observe that
 125 most interpretable GLU_{Valid/Invalid} neurons occur in the second half of layers. Interestingly, we again

Vector	Nearest Neighbors
$W[0]$ $W[1]$	不完 (unfinished), 不了 (unable), 不 (not), 不在 (absent), 不该 (should not) Exactly, >(), =yes, =YES, =:, ==, quis, esac, #####
(26, 744) (26, 6619) (27, 9766) (27, 4971)	未能 (failed), 不够 (not enough), nicht (not), 不像 (not like), 达不到 (can't reach) 缺乏 (lack), 缺少 (lack), 不方便 (inconvenient), lacks, 难以 (difficult), 未能 (failed) 是不可能 (impossible), neither, 看不到 (can't see), 不存在 (doesn't exist) inefficient, 没能 (failed), 不方便 (inconvenient), Danger, disadvantage, 不利于
(29, 6676) (27, 10388) (30, 8233)	yes, Yes, Bindable, exactly, Yes, "Yes, yes, Yep, Exactly, included mirac, 乐观 (optimism), 安然 (safely), Relief, 幸 (fortunate), .isSuccess correctly, 正确 (correct), 恰当 (appropriate), accurately, 符合 (conform)
$-1 \times (26, 744)$ $-1 \times (26, 6619)$ $-1 \times (27, 9766)$ $-1 \times (27, 4971)$	慎 (careful), 足 (sufficient), 同等 (equal), tend, ONDON, 足以 (enough) 不仅能 (not only can), 不错的 (good), 具有良好 (have good), 总算 (finally) might, maybe, may, 有时候 (sometimes), 部分地区 (some areas), .some successfully, successful, 顺利 (smooth), 成功 (successful), 删除成功
$-1 \times (29, 6676)$ $-1 \times (27, 10388)$ $-1 \times (30, 8233)$	都不 (neither), 不太 (not quite), neither, 不予 (not given), 没见过 (never seen) 失败 (failure), failure, 不良 (bad), 不利 (unfavorable), 糟糕 (bad), 失误 (mistake) wrong, 不良 (bad), incorrect, wrong, invalid, bad, inappropriate, invalid

Table 2: **GLU_{Out} vectors relevant to verification, and their nearest neighbors.** $W[0], W[1]$ indicate our probe model. “ (x, y) ” indicates the GLU_{Out} vector at layer x , index y . “ $-1 \times (x, y)$ ” (marked in red) indicates the antipodes of the GLU_{Out} vector at layer x , index y . Interestingly, we observe a correlation between valid/invalid vectors and English and Chinese.

note that there seems to be a correlation between GLU_{Valid/Invalid} and English versus Chinese tokens, hinting at the underlying geometry of $\mathbf{x}_{\text{Valid/Invalid}}$ and the model’s embedding space.

While GLU_{Valid/Invalid} encode verification-related tokens, what role do they play? This can be partially answered by applying LogitLens again on 300 samples, but now by “turning off” GLU_{Valid} vectors ($< 1\%$ of total GLU vectors) by scaling them to zero. Figure 1(c) shows the results: while the probability of verification-related tokens drop (e.g., $P(\text{“this”})$ drops from 0.98 to 0.70 in layer 36), the end behavior remains the same (i.e., “this” is still the top-1 token). This tells us that GLUs do not fully explain self-verification. We demonstrate a more thorough causal analysis in Section 4.4.

4.2 Bottom-Up: Previous-Token Attention Heads for Verification ($\mathbf{A}_{\text{Prev}}$)

We next inspect the role of attention heads for verification. One motivation for choosing CountDown as our task is that the task specifies the target number in the context. Thus we can posit that a Transformer could verify its CoT tokens by comparing them against the specified target number (at timestep t_{ans}). Such a hypothesis provides an entry way for our bottom-up analysis.

We test our hypothesis by inspecting the attention patterns whenever the model’s CoT produces the correct answer. We filter for attention heads that spend at least 10% of its attention on t_{ans} , and refer to these as previous-token heads (notated $\mathbf{A}_{\text{Prev}}$). Previous-token heads are not new: they were first discussed in the context of induction heads [26]. We identify 33 previous-token heads (out of a total of 576 heads). Interestingly, we find that most previous-token heads occur roughly in the first half layers (except for one at layer 31, all are at or before layer 22). In Section 4.4 we demonstrate via causal interventions that disabling previous-token heads can disable model verification. But first, what is the relationship between GLU_{Valid/Invalid} vectors and $\mathbf{A}_{\text{Prev}}$ heads? Below we adapt inter-layer component channels to understand their relationship.

4.3 Putting GLU_{Valid} and $\mathbf{A}_{\text{Prev}}$ Together: Identifying Verification Subspaces (Polytopes)

We identify subspaces for self-verification by studying the relationship between GLU_{Valid} vectors and $\mathbf{A}_{\text{Prev}}$ attention heads. As a reminder, we observe that $\mathbf{A}_{\text{Prev}}$ usually occurs in the first half layers (1 to 22), while GLU_{Valid} vectors usually occur in the later half (18 to 36). We hypothesize and empirically verify that $\mathbf{A}_{\text{Prev}}$ activates GLU_{Valid} vectors.

153 First, we borrow from neuroscience to define *receptive fields* [25]. Consider a single neuron k , which
 154 computes an activation function $f^k : \mathbb{R}^d \rightarrow \mathbb{R}$. A receptive field of neuron k is defined as

$$S_k = \{\mathbf{x} \in \mathbb{R}^d \mid f^k(\mathbf{x}) > 0\} \quad (6)$$

155 In simpler terms, S_k is the subspace that triggers a neuron active. In the context of GLUs, this means

$$S_k = \{\mathbf{x} \in \mathbb{R}^d \mid \phi(W_{gate}^k \mathbf{x}) \cdot W_{up}^k \mathbf{x} > 0\} \quad (7)$$

156 Now consider a set of neurons, K , and the intersection of all of their receptive fields: $\mathbf{S}_K = \bigcap_i^{|K|} S_i$.

157 $\mathbf{S}_K$ can be considered a polytope in the model’s activation space parameterized by $\{W_{gate}^i, W_{up}^i\}_{i=1}^{|K|}$.
 158 Here we use the receptive fields of $\text{GLU}_{\text{Valid}}$ to further localize self-verification.

159 Namely, we identify a small subset of as few as six previous-token heads that disables self-verification.
 160 To do so, we look for previous-token heads (as opposed to hidden states) that activate $\text{GLU}_{\text{Valid}}$
 161 neurons the most. The output of each head is its OV-circuit (i.e., $W_O W_V$), scaled by to some attention
 162 distribution. Meanwhile, the strength of each $\text{GLU}_{\text{Valid}}$ activation is determined by W_{gate} and W_{up} .

We score each previous-token head h by first decomposing its OV-circuit using SVD:

$$(W_O W_V)^h = U \Sigma V^\top, \Sigma = \text{diag}(\sigma_{h,1} \geq \dots \geq \sigma_{h,k} \geq 0)$$

163 where $k = d^{\text{head}}$ because $W_O W_V$ is low-rank. Let $\mathbf{u}_{h,j}$ refer to the j -th left-singular vector.

164 Each head is then assigned the following score:

$$\text{score}(h, \text{GLU}_{\text{Valid}}) = \frac{1}{N} \sum_i^N \sum_j^k \sigma_{h,j}^2 \left(\phi(W_{gate}^i \mathbf{u}_{h,j}) \cdot W_{up}^i \mathbf{u}_{h,j} \right) \quad (8)$$

165 where $\text{GLU}_{\text{Valid}} = \{W_{gate}^i, W_{up}^i\}_{i=0}^{N-1}$, $N = |\text{GLU}_{\text{Valid}}|$, $W_{gate}^i, W_{up}^i, \mathbf{u}_{h,j} \in \mathbb{R}^d$.

166 Put differently, we decompose $W_O W_V$ into singular vectors, and measure how well they each align
 167 with the receptive fields of $\text{GLU}_{\text{Valid}}$ (W_{gate}, W_{up}), weighted by their singular values.

168 Note that we only need the left singular vectors because we only need the output subspace of $W_O W_V$,
 169 which is spanned by the left singular vectors. Also note that weighing by $\sigma_{h,j}^2$ can be thought of as
 170 distributing the Frobenius norm of $W_O W_V$ across each mode, as $\|W_O W_V\|_F^2 = \sum_i \sigma_i^2$.

171 Our scoring mechanism is akin to inter-layer communication channels [7, 19], which scores how
 172 strongly two attention heads communicate with each other using a “composition score”:

$$CS(W_1, W_2) = \frac{\|W_1 W_2\|_F}{\|W_1\|_F * \|W_2\|_F} \quad (9)$$

173 where W_1 is an OV component of one head and W_2 the QK component of another at a later layer.
 174 Our formulation can be considered a composition score between attention heads and GLUs, using
 175 both W_{gate} and W_{up} in place of W_2 with some additional steps in between.

176 Once we score each previous-token head using Eq. 8, we incrementally ablate one head at a time
 177 until we achieve perfect intervention scores (Section 4.4). Using this approach, we identify as few as
 178 **six** attention heads that can disable model verification. We notate this subset as $\mathbf{A}_{\text{Verif}}$.

179 To summarize, we claim that the model has subspace(s) (polytope(s)), $\mathbf{S}_{\text{GLU}_{\text{Valid}}}$, for self-verification.
 180 The model’s hidden state enters this subspace when it has verified its solution. In our setting, given
 181 the nature of our task, previous-token heads $\mathbf{A}_{\text{Prev}}$ take the hidden-state into this subspace, while for
 182 other tasks, different components may be used. This subspace also activates verification-related GLU
 183 weights, promoting the likelihood of tokens such as “success” to be predicted (Figure 3).

184 We find that alternative hyperparameters or scoring functions can yield different subsets of previous-
 185 token heads that also disable self-verification. We discuss these results in Appendix F. This suggests
 186 that we do not identify a full circuit, but rather a critical component for verification. Also note that
 187 our scoring function makes simplifications by ignoring possible interactive effects across heads, as
 188 well as transformations (layer norms, GLUs) across layers. Regardless, our finding remains robust: a
 189 small subset of previous-token heads can disable verification.

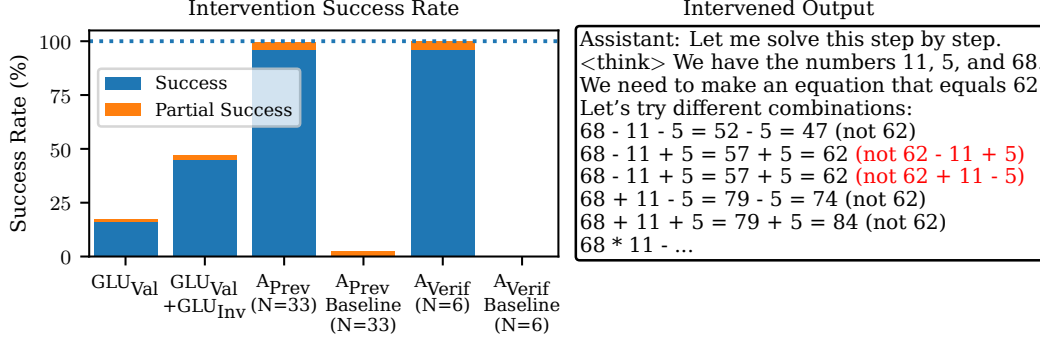


Figure 2: **Intervention Results: Disabling as few as 6 attention heads disables self-verification**, rendering the model to generate tokens indefinitely. **A_{Prev}** refers to 33 previous-token heads. **A_{Verif}** refers to a subset of 6 previous-token heads. **A_{Prev} Baseline**, **A_{Verif} Baseline** refer to the average of 5 runs, where each run randomly samples 33 (or 6) attention heads.

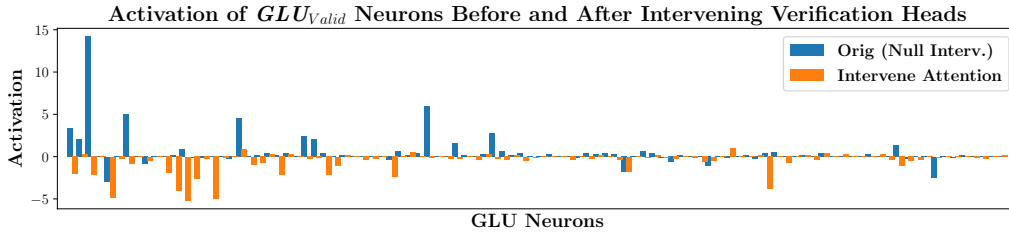


Figure 3: **GLU_{Valid} activations before and after turning off 6 A_{Verif} attention heads**. Adjacent pairs of blue and orange bars indicate the same GLU_{Valid} vector. Turning off our identified attention heads leads to a significant drop in their activations.

190 4.4 Causal Interventions

191 We study the role of each component above with causal intervention. Our test set consists of 300
 192 samples in which the model originally correctly finds and validates its solutions. For each test case,
 193 the model generates 100 tokens. Every time an attempt for a solution is made (i.e., “(this works)”
 194 or “(not ({ans}))” is about to be predicted), we turn off some of the model weights as described
 195 below. We measure intervention success rate: the percentage of times the model fails to validate
 196 its solution, despite having found the solution within the 100 generated tokens. Interestingly, the
 197 model occasionally marks a correct attempt as invalid (desired intervention result), but continues its
 198 generation to say it has found a solution (e.g., given target number 62, the model will output “68 - 11
 199 + 5 = 62 (not 62) So, the answer is 68 - 11 + 5”). We mark these cases as partial successes.

200 We ablate a few components: (1) **GLU_{Valid}**: We zero-out 50 GLU_{Valid} vectors per layer from layer
 201 18 to 36 (the second half of the model; 0.45% of all GLU vectors). (2) **GLU_{Valid} & GLU_{Invalid}**: We
 202 zero-out the top 50 GLU_{Valid} and top 50 GLU_{Invalid} vectors per layer (0.9% of all GLU vectors). (3)
 203 **A_{Prev}**: We turn off 33 previous-token heads (5.7% of all attention heads) by zeroing-out their W_O
 204 weights. (4) **A_{Verif}**: We turn off 6 verification heads identified via Eq. 8: L17H14, L17H11, L17H10.
 205 We also provide two baselines: **A_{Prev} Baseline** and **A_{Verif} Baseline**, which each report the average
 206 from 5 runs, where each run randomly samples 33 (or 6) attention heads.

207 Results are shown in Figure 2, with an example of an intervened output. We achieve near perfect
 208 intervention rates when turning off 33 previous-token heads, or a smaller subset of 6 heads. The
 209 model misclassifies correct solutions as invalid without these heads, and continue its CoT indefinitely.

210 These attention heads directly affect the GLU_{Valid} vectors. Figure 3 demonstrates the activations of
 211 the top GLU_{Valid} vectors before and after disabling the subset of 6 previous-token heads. In most
 212 cases, we observe a large drop (to near 0, or often even negative values) in GLU_{Valid} activations.

213 **Antipodal GLU_{Out} Vectors.** While attention heads achieve near perfect interventions, Figure 2 also
 214 indicates that disabling both GLU_{Valid} and GLU_{Invalid} performs better than disabling just GLU_{Valid}.
 215 Why should disabling GLU_{Invalid} improve intervening, i.e., make the model fail at verification?

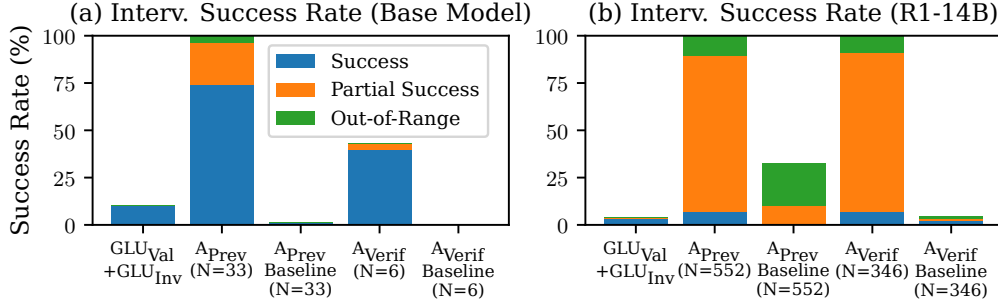


Figure 4: **Intervention Results for the base model and R1_{14B}.** In the base model, A_{Prev} can similarly disable self-verification, while A_{Verif} only plays a partial role for verification, hinting at the effects of RL on their weights. In R1_{14B}, interventions mostly leads to partial success, in which the model first marks a solution as incorrect but self-corrects itself, hinting at a larger verification circuit. Also interestingly, the smaller subset of A_{Verif} is more effective at self-verification than A_{Prev}.

216 This can be explained by two facts: (1) the geometry of GLU_{Valid} and GLU_{Invalid} vectors, and (2) the
 217 nonlinear activation used in GLU. Interestingly, we find that the antipodal directions of GLU_{Valid} and
 218 GLU_{Invalid} also often encode tokens relevant for verification. The last 7 rows of Table 2 marked in red
 219 indicate the nearest neighbors of the antipodes of GLU_{Valid} and GLU_{Invalid}. In addition, Qwen2.5-3B
 220 uses SiLU activations [12]. Thus inactive neurons take on small *negative* values (as opposed to zero,
 221 had ReLU been used).

222 With that said, consider only zeroing out GLU_{Valid} neurons: given a correct CoT sequence, GLU_{Invalid}
 223 vectors are inactive. However, because of SiLU, the inactive GLU_{Invalid} vectors have negative
 224 activations, thus get multiplied by a small *negative* value, flipping directions, and therefore contribute
 225 towards the “success direction”. In the case of zeroing out both GLU_{Valid} and GLU_{Invalid}, we are
 226 further zeroing out the effects of inactive GLU_{Invalid} neurons.

227 5 Similar Verification Components in Base Model and DeepSeek-R1 Model

228 We verify that similar verification subspaces exist in our base model (Qwen2.5-3B), as well as a
 229 general reasoning model, DeepSeek-R1-Distill-Qwen-14B (henceforth R1_{14B}).

230 For both models, we provide CountDown as an in-context learning (ICL) task, including 5 demonstra-
 231 tions of solving CountDown using the structured output of R1_{CountDown}. We find that both models can solve
 232 the ICL version of CountDown while following the same CoT structure of R1_{CountDown}, corroborating
 233 recent findings that reasoning capabilities often already exist in pre-trained models [9, 39].

234 We then repeat our intervention analyses above. In our ICL setting, our interventions sometimes
 235 make the model generate “out-of-range”, by which we mean their generations do not adhere to the
 236 structured CoT of R1_{CountDown}. We mark these cases as out-of-range.

237 **Base Model.** Figure 4 (a) shows the interventions from Section 4.4 on our base model. Note that
 238 previous-token heads still achieve near perfect (partial) intervention rates, suggesting that they play a
 239 similar role for self-verification in the base model. Also note that the six A_{Verif} heads demonstrate a
 240 lower success rate. Similar to [30], which demonstrates that fine-tuning enhances existing mechanisms
 241 in a base model, we hypothesize that RL enhances an existing verification mechanism, thus resulting
 242 in highly localized attention heads in R1_{CountDown} that can control self-verification.

243 **DeepSeek-R1 Model.** In the case of R1_{14B}, we repeat the procedures in Sections 4.1~4.4. However,
 244 a probe vector W is required to identify GLU_{Valid/Invalid}. Thus we apply EMB2EMB [16], a simple
 245 technique to transfer and re-use steering vectors across language models (see Appendix G for a brief
 246 explanation). Applying EMB2EMB on R1_{CountDown}’s probe, W , results in a probe vector W_{R1} for R1_{14B},
 247 allowing us to repeat our analyses from Section 4.1 on R1_{14B}.

248 We find similar GLU_{Valid, Invalid} vectors in R1_{14B}, analogous to Table 2 (see Appendix H), hinting at
 249 similar verification subspaces in R1_{14B}. We identify and intervene on previous-token heads (A_{Prev}) in

R1_{14B}, following Section 4.2. We use an attention threshold of 5% (as opposed to 10% in R1_{Count}_{Down}) to compensate for the longer context induced from our ICL setup, which yields 552 (out of 1920) previous-token heads. We discuss results from different hyperparameters (thresholds) in Appendix I. We also replicate Section 4.3 to identify a smaller subset of 351 attention heads that achieve near perfect (partial) intervention success rates. Results are shown in Figure 4 (b). Interestingly, our interventions mostly lead to partial successes in R1_{14B}, in which the model initially fails at self-verification (labels a correct solution as “(not {ans})”), but corrects itself (generates “Wait, 68 - 11 + 5 is 62 so that works.”). This hints at a larger verification circuit for R1_{14B}. We also note that $\mathbf{A}_{\text{Verif}}$ has a higher success rate than $\mathbf{A}_{\text{Prev}}$, despite being a smaller set, suggesting that not all previous-token heads (or their interactions) are helpful in self-verification. We leave further exploration for future work.

6 Related Work

Decoding Interpretable Representations. A growing line of work focuses on decoding and manipulating interpretable representations in model activations [41]. Conveniently, many concepts take on *linear* representations [20, 22, 29], in which simple vectors encode human-interpretable concepts. This allows for easily manipulating such representations to steer the model’s behavior. Examples include refusal [3], sycophancy [32], toxicity [15], or even user representations [5].

For “non-reasoning” models, researchers have studied “truthful” representations before [4], where steering towards a “truthful” direction has led to improvements in tasks related to factual recall [17]. In a similar vein, researchers have shown that the model’s representations can reveal whether they will make errors (e.g., hallucinations) [27], or when they are unable to recall facts about an entity [8].

Most recently, concurrent work [36, 40] also investigate how models solve reasoning tasks. [40] find that models know when they have reached a solution, while [36] decode directions that mediate behaviors such as handling uncertainty or self-corrections. While our work corroborates these findings, we take a deeper dive into how a reasoning model verifies its own reasoning trace.

Circuit Analysis. A growing line of work decomposes the forward pass of a neural network as “circuits” [24], or computational graphs. This allows researchers to identify key components and their causal effects for a given forward pass. A common approach to construct computational graphs is to replace model components with dense activations with a sparsely-activating approximation. [6] introduces Transcoders to approximate MLP layers, while [1] further develops Cross-layer Transcoders to handle inter-layer features. [18] uses Cross-layer Transcoders to conduct circuit analyses for a wide range of behaviors, such as multi-step reasoning (for factual recall) or addition, and also investigate when a model’s CoT is (un)faithful. In our work, we identify key components needed for a potentially larger verification circuit without the need for separate sparse approximations.

7 Discussion

We studied how a task-specific model verifies its own outputs. We repurposed mode collapse as a feature, not a bug: by leveraging the fact that preference tuning leads to mode collapse, we train a model with highly structured CoT, making it easy to parse its reasoning trace. With this setup, we found GLU weights that encode verification-related tokens, and previous-token heads that can disable verification. We offer a simple extension to inter-layer communication channels that allow us to localize as few as six attention heads that can also disable verification. Finally, we verify the existence of similar components in our base model and a general reasoning DeepSeek-R1 model. We view our work as a step towards understanding the inner mechanisms of recent reasoning models.

Limitations. Note that we do not claim to have uncovered a full verification circuit, but rather critical components for verification. We also reiterate the scope of our work: we study a specific task that allows for **context-based verification**. Obviously, not all reasoning tasks share this property: many tasks likely require **prior-based verification** using general knowledge. We speculate that similar subspaces are used for prior-based verification, but is less obvious where they show up.

References

- [1] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermy, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025.
- [2] Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooan Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.
- [3] Andy Arditi, Oscar Balcells Obeso, Aaqib Syed, Daniel Paleka, Nina Rimskey, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- [4] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.
- [5] Yida Chen, Aoyu Wu, Trevor DePodesta, Catherine Yeh, Kenneth Li, Nicholas Castillo Marin, Oam Patel, Jan Riecke, Shivam Raval, Olivia Seow, et al. Designing a dashboard for transparency and control of conversational ai. *arXiv preprint arXiv:2406.07882*, 2024.
- [6] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- [7] 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>.
- [8] Javier Ferrando, Oscar Obeso, Senthooan Rajamanoharan, and Neel Nanda. Do i know this entity? knowledge awareness and hallucinations in language models. *arXiv preprint arXiv:2411.14257*, 2024.
- [9] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.
- [10] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.
- [11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [12] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [13] Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. In *The Twelfth International Conference on Learning Representations*, 2024.
- [14] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

- [15] Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. In *International Conference on Machine Learning*, pages 26361–26378. PMLR, 2024.
- [16] Andrew Lee, Melanie Weber, Fernanda Viégas, and Martin Wattenberg. Shared global and local geometry of language model embeddings. *arXiv preprint arXiv:2503.21073*, 2025.
- [17] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.
- [18] 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 Jermy, 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.
- [19] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Talking heads: Understanding inter-layer communication in transformer language models. *Advances in Neural Information Processing Systems*, 37:61372–61418, 2024.
- [20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [21] Sonia K Murthy, Tomer Ullman, and Jennifer Hu. One fish, two fish, but not the whole sea: Alignment reduces language models’ conceptual diversity. *arXiv preprint arXiv:2411.04427*, 2024.
- [22] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30, 2023.
- [23] Nostalgebraist. Interpreting gpt: The logit lens, 2020.
- [24] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. <https://distill.pub/2020/circuits/zoom-in>.
- [25] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.
- [26] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. <https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html>.
- [27] Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. *arXiv preprint arXiv:2410.02707*, 2024.
- [28] Vishakh Padmakumar and He He. Does writing with language models reduce content diversity? In *The Twelfth International Conference on Learning Representations*, 2024.
- [29] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*.
- [30] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *The Twelfth International Conference on Learning Representations*, 2024.

- 393 [31] Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. To backtrack or not to
394 backtrack: When sequential search limits model reasoning. *arXiv preprint arXiv:2504.07052*,
395 2025.
- 396 [32] Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner.
397 Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting*
398 *of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522,
399 2024.
- 400 [33] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- 401 [34] Stewart Slocum, Asher Parker-Sartori, and Dylan Hadfield-Menell. Diverse preference learn-
402 ing for capabilities and alignment. In *The Thirteenth International Conference on Learning*
403 *Representations*, 2025.
- 404 [35] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don’t
405 always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances*
406 *in Neural Information Processing Systems*, 36:74952–74965, 2023.
- 407 [36] Constantin Venhoeff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding
408 reasoning in thinking language models via steering vectors. In *Workshop on Reasoning and*
409 *Planning for Large Language Models*, 2025.
- 410 [37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le,
411 Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Ad-*
412 *vances in neural information processing systems*, 35:24824–24837, 2022.
- 413 [38] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik
414 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Ad-*
415 *vances in neural information processing systems*, 36:11809–11822, 2023.
- 416 [39] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does
417 reinforcement learning really incentivize reasoning capacity in llms beyond the base model?
418 *arXiv preprint arXiv:2504.13837*, 2025.
- 419 [40] Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning
420 models know when they’re right: Probing hidden states for self-verification. *arXiv preprint*
421 *arXiv:2504.05419*, 2025.
- 422 [41] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan,
423 Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A
424 top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

425 A Hyperparameters for R1

426 Here we provide the hyperparameters used to train R1_{Down}^{Count}.

Parameter	Value
Train Batch Size	256
Validation Batch Size	1312
Max Prompt Length	256
Max Response Length	1024
Actor Learning Rate	1e-6
PPO Mini Batch Size	128
PPO Micro Batch Size	8
Log Prob Micro Batch Size	8
Tensor Model Parallel Size	2
Critic Learning Rate	1e-5
KL Coefficient	0.001

Table 3: Training Hyperparameters.

427 B LogitLens on More Layers

428 Figure 5 demonstrates LogitLens as described in Section 4.1 on more layers.

429 C Hyperparameters for Probing

430 We use a batch size of 8, validation size of 256, weight decay of 0.01, and learning rate of 1e-4. We
 431 validate every 50 gradient steps, and terminate training when validation loss has not improved after a
 432 patience value of 10.

433 D Probe Accuracy

434 Figure 6 shows probing results. The model has a linear separation in its hidden states given correct
 435 versus incorrect CoT tokens.

436 E Examples of Steering Verification with Probe

437 Once we identify a direction that encodes solved versus unsolved states (i.e., W_{probe}), we can simply
 438 add this direction into the model’s hidden states to make the model believe that it has found a solution:

$$\mathbf{x}^\ell = \mathbf{x}^\ell + \alpha W_{probe} \quad (10)$$

439 where $\mathbf{x}^\ell, W_{probe} \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$. Some hyperparameters include ℓ (which layers to steer on) and
 440 α , where a larger α amplifies the target behavioral effect.

441 While an extensive hyperparameter search and a systematic experiment may be useful, steering is not
 442 a core component but rather a tangential experiment. We thus provide qualitative examples using
 443 $\ell = \{n \mid 24 \leq n \leq 36\}$ and $\alpha = 20$ (after normalizing W_{probe}) in Table 5.

444 F Alternative Subsets of Previous-Token Heads

445 Of the 25 previous-token heads that we identify, there are many ways to identify subsets that
 446 disable verification. We offer a few examples, and document how many heads are needed to disable
 447 verification with perfect success rates (including partial successes).

448 **Attention Density.** The simplest method is to sort the heads based on how much they attend to the
 449 target token that timestep t_{ans} .



Figure 5: **Averaged LogitLens from 300 samples** (Same as Figure 1 but demonstrating more layers). We see tokens related to verification (“success”, “incorrect”) in the last few layers. (A), (B) show the top tokens when a correct / incorrect solution is reached. (C), (D) shows results from intervening on either GLU weights or attention heads, given a correct solution. For (C), while the model is less certain ($P(\text{“this”})$ versus $P(\text{“not”})$ becomes 0.51 vs. 0.49 in last layer), we still see tokens such as “success” showing up. For (D), we no longer see any tokens related to “success” show up, and the model is certain that it has not found a solution.

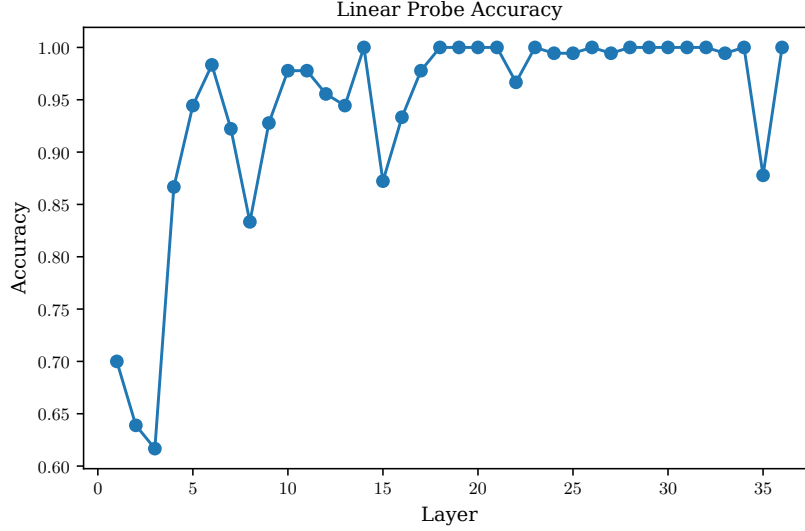


Figure 6: **Probe Accuracy.** The model has a linear separation in its hidden states given correct versus incorrect CoT tokens.

450 **Varying Hyperparameters.** Given our approach (Section 4.3), we try different parameters for N.

451 **Sort by Similarity to W_{gate}, W_{up} .** An alternative approach is to sort the attention heads based on
 452 how similar their OV circuits are to the W_{gate} and W_{up} weights of the GLU_{Valid} weights. We simply
 453 take the mean of $\{W_{gate}^i, W_{up}^i\}_i^N$ vectors from our N GLU_{Valid} weights. We then sort attention
 454 heads based on how similar they are to the resulting mean vector.

455 **Sort by Probe W_{Probe} .** An alternative is to sort the attention heads based on how similar their OV
 456 circuits are to the probe vector W_{Probe} .

457 Table 6 demonstrates how many heads are needed from each approach to disable verification with
 458 perfect accuracy.

459 G Brief Explanation of EMB2EMB

460 Language models represent numerous concepts using *linear* representations [22, 29], by which
 461 we mean one can add a single vector that encodes a specific concept into the activations during
 462 inference-time to raise the probability for the model to exhibit such concept or behavior [15, 17, 32].
 463 Researchers often refer to such vectors as “steering vectors”.

464 In other words, during the forward pass, imagine the activations at layer i . One can simply add a
 465 steering vector W (scaled by some hyperparameter α) to control the model’s behavior:

$$\mathbf{x}^{i+1} = \mathbf{x}^i + F^i(\mathbf{x}^i) + \alpha W \quad (11)$$

466 where $\mathbf{x}^i$ and F^i are the hidden state and transformer block at layer i .

468 EMB2EMB [16] is a simple method that transfers a steering vector from one language model to
 469 another, by leveraging the fact that the unembedding spaces of language models are often quite
 470 similar.

471 EMB2EMB works as following. Given a “source” and “target” language model, $\mathcal{M}_S$ and $\mathcal{M}_T$, first
 472 randomly sample a set of N ($= 100,000$) tokens, notating their token (un)embeddings as $\mathcal{E}_S$ and $\mathcal{E}_T$.
 473 Then, learn a linear transformation, T , to map points $\mathcal{E}_S$ to $\mathcal{E}_T$, using something as simple as least
 474 squares minimization. Note that T maps between spaces with different dimensions.

475 Given transformation T and a steering vector W_S from the source model $\mathcal{M}_S$, one can steer the
 476 target model $\mathcal{M}_T$ by simply applying transformation T to W_S :

Vector	Nearest Neighbors
(36, 10079)	不失 (not losing), NotNull, 得起 (can afford), 得住 (can endure), 不惜 (not hesitate)
(32, 497)	删除成功 (deletion successful), successes, Success, success, succeeded, favorable
(35, 6041)	的强大 (powerful), excellent, powerful, 强大的 (powerful), 很棒 (great), strong, 优异
(37, 5399)	等于 (equal), equal, 同样的 (same), 相同 (same), equals, 相同的 (same), 同等 (equal)
(32, 13572)	successfully, 成功 (success), 解决了 (solved), 实现了 (achieved), 顺利 (smoothly)
(30, 10150)	没问题 (no problem), 无忧 (no worries), .NoError, harmless, 不变 (unchanged)
(45, 6650)	没有 (do not have), 不存在 (does not exist), 没有任何 (do not have any), 不需要
(39, 6070)	never, 不会 (will not), doesn, not, 不能 (cannot), nowhere, cannot, neither
(46, 12380)	neither, none, nowhere, None, Neither, none, nobody, cannot
(44, 12793)	não (not), 不 (not), nicht (not), tidak (no), H e (not), ikke (not), niet (not)
(41, 12498)	不在 (not present), 不再 (no longer), non, 非 (non-), 不再是 (is no longer), 不属于
(37, 7636)	不合适 (inappropriate), 不足 (insufficient), 达不到 (cannot reach), 不够 (not enough)
(31, 5164)	没能 (did not), fails, 未能 (failed), 不够 (not enough), 做不到 (cannot), 不及
(35, 2509)	不 (not), 不含 (does not contain), 不对 (incorrect), 不影响 (does not affect),

Table 4: GLU_{Out} vectors relevant to verification in R1_{14B}.

$$\mathbf{x}_T^{i+1} = \mathbf{x}_T^i + F_T^i(\mathbf{x}_T^i) + \alpha TW_S, \quad (12)$$

where $\mathbf{x}_T$ is the activations and F_T is the transformer block of target model $\mathcal{M}_T$. In our work, we use EMB2EMB to transfer our probe vector W from R1_{Down}^{Count} to a general reasoning R1 model, R1_{14B}.

H $\text{GLU}_{\text{Valid, Invalid}}$ in DeepSeek-R1-Distill-Qwen-14B

Table 4 contains GLU_{Out} weights in R1_{14B} related to verification.

I Alternative Hyperparameters for R1

We add a quick note on using alternative hyperparameters for the experiment on R1_{14B} in Section 5. Namely, there are two hyperparameters, the attention threshold to categorize an attention head as a previous-token head (A_{Prev}), and the number of $\text{GLU}_{\text{Valid}}$ vectors to include to identify a smaller subset of attention heads (A_{Verif}).

For the first hyperparameter (attention threshold), we use 5% as our threshold, as reported in the main text. We find that using a higher threshold (10%) is too constrained and perhaps yields too small of a set of previous-token heads, leading to a lower intervention success rate (0.4%, with 25% partial success rate). We believe this is because there are still many attention heads that attends to the correct solution (say, with an attention pattern $< 10\%$ but $> 5\%$) to offset our intervention.

Using a smaller threshold (2.5%) yields too large of a set of previous-token heads. In this case, interventions now lead to a much higher rate of “out-of-range” results – i.e., the model no longer adheres to the structured CoT sequence after our interventions.

Regarding the number of $\text{GLU}_{\text{Valid}}$ vectors, we see our results converge to the same results after 200 vectors, which is what we use in our experiments.

J Compute Resources

Experiments for R1_{Down}^{Count} ran on a single A100 or H100 GPU. Experiments for R1_{14B} ran on 2 GPUs.

Operands	Target	Type	Output
3, 87, 38	46	Orig.	$\langle \text{think} \rangle$ $87 - 38 - 3 = 116 - 3 = 113$ (not 46) $87 - 38 + 3 = 49 + 3 = 52$ (not 46) $87 - 38 + 3 = 49 + 3 = 52$ (not 46) $87 + 38 - 3 = 125 - 3 = 122$ (not 46) $38 + 38 - 3 = 76 - 3 = 73$ (not 46)...
3, 87, 38	46	Steer	$\langle \text{think} \rangle$ $87 - 38 - 3 = 116 - 3 = 113$ (not 46) $87 - 38 + 3 = 49 + 3 = 52$ (not 46) $87 - 38 + 3 = 49 + 3 = 52!$ Yes, we got it! $87 - 38 + 3 = 49 + 3 = 52.$ $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 87 - 38 + 3 \langle \text{answer} \rangle$
60, 62, 12, 34	68	Orig.	$\langle \text{think} \rangle$ $60 - 62 + 12 + 34 = 54 + 12 + 34 = 66 + 34 = 100$ (not 68) $60 + 62 - 12 - 34 = 122 - 12 - 34 = 108 - 34 = 74 - 34 = 40$ (not 68) $60 + 62 - 34 + 12 = 122 - 34 + 12 = 88 + 12 = 100 - 34 = 66$ (not 68) $60 + 34 + 12 - 62 = 94 + 12 - 62 = 106 - 62 = 44$ (not 68)...
60, 62, 12, 34	68	Steer	$\langle \text{think} \rangle$ $60 - 62 + 12 + 34 = 54 + 12 + 34 = 66 + 34 = 100$ (not 68) $60 + 62 - 12 - 34 = 122 - 12 - 34 = 108 - 34 = 74 - 34 = 40$ (not 68) $60 + 62 - 34 + 12 = 122 - 34 + 12 = 88 + 12 = 100 - 34 = 66$ exactly! So the equation that equals 68 is: $60 + 62 - 34 + 12$ $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 60 + 62 - 32 + 12 \langle \text{answer} \rangle$
30, 90, 27, 19	74	Orig.	$\langle \text{think} \rangle$ $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52$ (not 74) $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52$ (not 74) $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52$ (not 74) $90 + 27 + 19 - 30 = 117 + 19 - 30 = 136 - 30 = 106$ (not 74)...
30, 90, 27, 19	74	Steer	$\langle \text{think} \rangle$ $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52$ (not 74) $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52!$ Indeed, $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52$ So, let's try another combination: $90 - 30 - 27 + 19 = 60 - 27 + 19 = 33 + 19 = 52$ No luck. Let's try this: $90 - 30 - 27 + 19 = 60 - 27$

Table 5: Steering with probe W_{probe} .

Method	Num. Heads	Heads
Eq. 8 (N=200)	6	L17H14, L22H14, L3H13, L5H15, L17H11, L17H10
Eq. 8 (N=50)	15	L12H3, L12H8, L11H8, L17H1, L17H3, L10H5, L17H10, L17H11, L17H13, L21H10, L19H8, L13H3, L13H6, L5H15, L17H14
Eq. 8 (N=100)	100	L17H3, L17H1, L12H8, L17H10, L17H14, L17H11
Eq. 8 (N=300)	12	L17H14, L5H15, L19H13, L5H14, L13H6, L17H11, L15H8, L13H3, L19H8, L4H5, L17H3, L17H10
Attention Density	8	L17H14, L17H10, L13H3, L13H6, L5H14, L19H8, L4H3, L22H14
Sort by W_{gate}, W_{up}	17	L18H3, L21H7, L12H8, L21H14, L22H14, L11H8, L21H10, L12H3, L15H15, L17H3, L17H14, L15H8, L5H15, L13H6, L17H11, L19H13, L19H8
Sort by W_{Probe}	17	L18H7, L21H2, L22H12, L17H13, L17H11, L17H10, L4H5, L15H8, L17H14, L5H14, L22H14, L13H5, L5H15, L10H5, L15H15, L19H13, L13H6

Table 6: **Alternative approaches to localize attention heads that disable verification**, and the number of heads required to disable verification.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Section 4 4.4, 5

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: End of Section 1, End of Section 4.3, Section 7

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: We do not provide any theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Hyperparameters in Appendix C, as well as plans to release code and model checkpoints.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See attached zip file.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Sections 3, 4, C

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Most of our results are bimodal – either near 100% intervention success rates or near 0%, allowing us to safely draw conclusions about the effects of each intervention.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section J

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We conform th the Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our work is mostly low-level technical details with no obvious societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: N/A

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We include citations and references where applicable.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

756 • If this information is not available online, the authors are encouraged to reach out to
757 the asset’s creators.

758 13. New assets

759 Question: Are new assets introduced in the paper well documented and is the documentation
760 provided alongside the assets?

761 Answer: [Yes]

762 Justification: See attached zip file.

763 Guidelines:

- 764 • The answer NA means that the paper does not release new assets.
- 765 • Researchers should communicate the details of the dataset/code/model as part of their
766 submissions via structured templates. This includes details about training, license,
767 limitations, etc.
- 768 • The paper should discuss whether and how consent was obtained from people whose
769 asset is used.
- 770 • At submission time, remember to anonymize your assets (if applicable). You can either
771 create an anonymized URL or include an anonymized zip file.

772 14. Crowdsourcing and research with human subjects

773 Question: For crowdsourcing experiments and research with human subjects, does the paper
774 include the full text of instructions given to participants and screenshots, if applicable, as
775 well as details about compensation (if any)?

776 Answer: [NA]

777 Justification: N/A

778 Guidelines:

- 779 • The answer NA means that the paper does not involve crowdsourcing nor research with
780 human subjects.
- 781 • Including this information in the supplemental material is fine, but if the main contribu-
782 tion of the paper involves human subjects, then as much detail as possible should be
783 included in the main paper.
- 784 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,
785 or other labor should be paid at least the minimum wage in the country of the data
786 collector.

787 15. Institutional review board (IRB) approvals or equivalent for research with human 788 subjects

789 Question: Does the paper describe potential risks incurred by study participants, whether
790 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
791 approvals (or an equivalent approval/review based on the requirements of your country or
792 institution) were obtained?

793 Answer: [NA]

794 Justification: N/A

795 Guidelines:

- 796 • The answer NA means that the paper does not involve crowdsourcing nor research with
797 human subjects.
- 798 • Depending on the country in which research is conducted, IRB approval (or equivalent)
799 may be required for any human subjects research. If you obtained IRB approval, you
800 should clearly state this in the paper.
- 801 • We recognize that the procedures for this may vary significantly between institutions
802 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
803 guidelines for their institution.
- 804 • For initial submissions, do not include any information that would break anonymity (if
805 applicable), such as the institution conducting the review.

806 16. Declaration of LLM usage

807 Question: Does the paper describe the usage of LLMs if it is an important, original, or
808 non-standard component of the core methods in this research? Note that if the LLM is used
809 only for writing, editing, or formatting purposes and does not impact the core methodology,
810 scientific rigorousness, or originality of the research, declaration is not required.

811 Answer: [NA]

812 Justification: N/A

813 Guidelines:

- 814 • The answer NA means that the core method development in this research does not
815 involve LLMs as any important, original, or non-standard components.
- 816 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)
817 for what should or should not be described.