The Unembedding Bottleneck: A Mechanistic Analysis of Single Digit Counting in LLMs

Satwik Sunnam[1] Raghav Magazine[1] Vatsalya Singh[2] Lavanya Kotha[3]
Xingjian Li[1,*] Min Xu[1,*]

[1]Carnegie Mellon University [2]IIT Guwahati [3]Independent Researcher

[*]Corresponding authors: xingjia2@andrew.cmu.edu, mxu1@cs.cmu.edu

Abstract

Large language models consistently fail at elementary counting tasks despite strong performance on complex reasoning benchmarks. Recent work has characterized counting failures behaviorally or identified circuits through which counting succeeds, but a mechanistic account of where and why counting breaks down remains absent. In this work, we provide such an account by tracing the count signal from embedding to output and identifying the geometric bottleneck that prevents correct readout for single digit counting problems. Using a dual-metric framework combining Signal-to-Noise Ratio (SNR) with linear probing, we show that counting information in models is linearly decodable through the final layer. The failure in counting is not caused by a loss of count information, but by limited accessibility i.e. the digit-token columns of the unembedding matrix W_U are orthogonal to the count subspace in the residual stream, rendering correctly computed counts unreadable at the output. We term this as the **Unembedding Bottleneck** and to causally validate it, we apply a **Probe-Informed LoRA** correction targeting W_U, which restores single-digit counting accuracy by up to 80% across six models while updating only $\sim 0.001\%$ of parameters and preserving general capabilities. Our findings reveal a broader class of failure in LLMs where information is internally present but geometrically inaccessible to the output projection.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks such as reasoning, coding and multi-modal understanding [Achiam et al., 2023, Chowdhery et al., 2023, Alayrac et al., 2022]. Despite these advances, their ability to perform reliable counting remains inconsistent and poorly understood [Dziri et al., 2023]. Prior work has shown that even state-of-the-art models struggle with seemingly simple counting tasks, particularly when inputs require compositional reasoning, long-range dependencies or robustness to noise Acharya et al. [2019]. This raises a fundamental question: *How do LLMs internally represent and compute counts ?* Recent research in mechanistic interpretability seeks to answer such questions by reverse-engineering neural networks into understandable computational components Olah et al. [2020]. Instead of treating models as black boxes, this line of work aims to identify circuits, attention patterns and neuron-level representations that correspond to specific algorithmic behaviors. While mechanistic interpretability has successfully uncovered structure in tasks such as induction, in-context learning [Olsson et al., 2022] and arithmetic Lee et al. [2023], comparatively little is known about how counting is implemented within these models.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In this work, we aim to bridge this gap by providing a mechanistic analysis of counting in LLMs. We investigate how information about counts is propagated across model layers and aim to answer two Research Questions (RQ) as described below:

- **RQ1**: How is the count signal for single-digit quantities represented in the residual stream of LLMs and how does this representation evolve across layers?
- **RQ2**: What underlying factors contribute to the failure of LLMs on seemingly trivial single digit counting tasks?

A wide range of mechanistic interpretability techniques have been developed to understand the internal representations of neural networks. Linear probes [Raimondi and Gabbrielli, 2026] extract linearly decodable features from intermediate representations, while feature visualization [Olah et al., 2020] and sparse autoencoders [Cunningham et al., 2023] identify interpretable components within learned representations. Causal intervention methods, including activation patching [Kramár et al., 2024, Meng et al., 2022] and circuit analysis [Elhage et al., 2021, Conmy et al., 2023], go further by testing whether specific components are functionally necessary for a model's computations. While these methods have yielded valuable insights, they occupy different points on a cost–informativeness tradeoff. Probing methods can recover linearly decodable features even when these features are distributed across many dimensions, redundant or not causally involved in the model's computation [Belinkov, 2022, Voita and Titov, 2020], potentially overstating a representation's functional utility. Causal methods establish functional dependence but require targeted, component-level interventions, making systematic layer-wise tracking of signal evolution expensive. Feature discovery approaches such as SAEs [Bricken et al., 2023a] may reveal interpretable components but do not directly quantify how robustly these components contribute to downstream predictions.

To diagnose how the count signal evolves across layers, we require a metric that captures not merely whether counting information is present, but how *strongly* it is expressed . Linear probes answer a binary question - can a learned classifier extract the count? - but remain agnostic to the geometric concentration of the signal i.e a probe can successfully decode a signal even when information is diffused across many dimensions. We use Signal-to-Noise Ratio (SNR) alongside linear probes to trace the count signal across layers. This dual measurement proves critical as we find that counting information persists in a distributed linear subspace throughout as probes decode the count well across layers yet the SNR on the dominant axis collapses after early layers, indicating that the signal disperses across multiple directions rather than remaining concentrated. Finally, count information remains linearly decodable up to the final layer, but lies in a subspace weakly aligned with the digit columns of W_U rendering it unreadable by W_U . This observation leads to our central finding: *counting failures in LLMs arise not from the absence of count information, but from a geometric misalignment between the count subspace and the unembedding matrix*. We use SNR along the dominant principal axis as it quantifies the strength of the count signal on a single direction, distinguishing layers where count identity dominates the variance from layers where it is buried beneath unrelated variation. This distinction is methodologically consequential as layers with high SNR concentrate the count signal into a low-dimensional, high-quality subspace that can be reliably estimated. We exploit this property by training linear probes to identify the count subspace and subsequently use the resulting probe directions to guide the alignment of W_U with that subspace.

Our contributions include the following :

- We identify the **Unembedding Bottleneck** as the root cause of counting failures i.e the digit-token columns of W_U are orthogonal to the count subspace in the residual stream, rendering correctly computed counts unreadable at the output.
- We causally validate the diagnosis using a **Probe-Informed LoRA** correction targeting W_U, which restores single digit counting accuracy by up to 80% across six models while updating only $\sim 0.001\%$ of parameters and preserving general capabilities of the model.
- We provide extensive experiments across multiple models and diverse counting task variations, including novel token pairs, semantic categories and prompt formulations, showing that our findings generalize across architectures and evaluation settings.

2 Related Work

2.1 Mechanistic Interpretability of Transformer Models

Mechanistic interpretability aims to reverse engineer neural networks by identifying the internal circuits and representations that give rise to model behavior. Early work introduced techniques such as feature visualization and circuit analysis [Elhage et al., 2021, Olah et al., 2020] to uncover interpretable structure in transformers, including attention head roles and neuron-level functions. Subsequent studies developed causal interventions [Wang et al., 2022] such as activation patching and path tracing to establish functional contributions of components to model outputs. More recent work has scaled these approaches to larger models, revealing distributed and compositional representations, sparse feature structures [Cunningham et al., 2023] and modular circuits underlying complex behaviors. Complementary lines of research also explore automated interpretability [Haklay et al., 2025, Conmy et al., 2023] and dictionary learning methods [Bricken et al., 2023b] to extract human-interpretable features from high-dimensional activations. These works establish mechanistic interpretability as a rigorous framework for understanding how transformer models compute, moving beyond behavioral analysis toward causal, circuit-level explanations [Geva et al., 2021].

2.2 Counting and Numerosity in Language Models

Despite strong performance on general reasoning tasks, both LLMs and vision-language models (VLMs) exhibit significant limitations in counting, which has been highlighted by recent benchmarks. [Guo et al., 2025] introduce a controlled compositional counting benchmark and shows that VLMs fail when multiple object types are present, even if they succeed in simpler settings. [Sengupta et al., 2025] demonstrate that counting performance degrades with increasing visual complexity, object count and that models fail to generalize beyond the range of numbers seen during training. To address these limitations, [Qharabagh et al., 2024] propose a divide-and-conquer strategy that decomposes counting into sub-counting problems, significantly improving generalization across datasets without additional training. These works collectively show that counting remains a brittle and poorly generalized capability in current LLMs.

2.3 Interpreting Counting Mechanisms in LLMs

Beyond evaluating performance, recent work seeks to understand how counting is internally represented and computed. Gurnee et al. [2026] show that counting can be realized through structured geometric transformations over feature manifolds, revealing distributed yet interpretable computational pathways. Che et al. [2026] extend mechanistic analysis to VLMs and find structured *counting circuits* using activation patching and attention-based analyses, demonstrating that counting behavior emerges from coordinated interactions across attention heads and layers. [Hasani et al., 2025] provide evidence that counting emerges as a structured, layer-wise process where early layers encode small counts and later layers represent larger counts. The study identifies an implicit counter mechanism that updates with each observed item and is often stored in the final token or visual region. [Hasani et al., 2026] show that counting is performed across layers in transformers and that failure to count longer sequences is a consequence of architectural constraints. [Yehudai et al., 2024] show that when embedding dimension is at least as large as the vocabulary, transformers can perform exact counting, whereas smaller embedding dimensions lead to embedding crowding which makes counting harder. Zhang et al. [2024] investigate the impact of tokenization strategies on counting, while Hou et al. [2025] study how prompting models differently affects their ability to count.

Our work differs from prior studies in a key respect. While existing literature either benchmarks counting failures behaviorally or characterizes how counting succeeds through structured circuits and geometric representations, we provide a mechanistic account of *where and why* counting breaks down. Specifically, we identify the unembedding matrix as a functional bottleneck in single-digit counting and validate this diagnosis causally through targeted parametric repair.

3 Methodology

We investigate counting failures by tracing the count signal from embedding to output and identifying the bottleneck that prevents correct readout. Our methodology has three components: (i) mea-

Counting Prompt Format

```
Given the sequence M M N M M N N M N, how many times does N appear?  4
Given the sequence N M M M M M M M M, how many times does N appear?  1
Given the sequence N N N M N N N N N, how many times does N appear?  8
Given the sequence M M N N M N N M N, how many times does N appear?
```

Figure 1: **Canonical counting prompt format.** A 3-shot template instructing the model to count occurrences of a target token. The appear token marks the position at which the counting task is fully specified and the ? is the generation token

suring the count signal with SNR and linear probes, (ii) quantifying the unembedding bottleneck, (iii) causally validating the diagnosis through targeted LoRA repair.

3.1 Measuring the Count Signal

To quantify how cleanly the counting signal separates from noise at each layer, we define a Signal-to-Noise Ratio (SNR) based on the Fisher discriminant ratio along the dominant principal axis. For each layer we collect the residual-stream activations $X \in \mathbb{R}^{n \times d}$ at the token position of interest, where n is the number of samples and d is the hidden-state dimensionality. We first do PCA (Principal Component Analysis) and project every sample $x_i \in X$ onto the first principal component $\mathbf{v}_1$, obtaining scalar projections z_i. To measure whether the variance along $\mathbf{v}_1$ is due to count identity rather than within-class noise, we compute the between-class and within-class variances of the projections. Let $\mathcal{C}$ denote the set of count classes and $\bar{z}_c = \frac{1}{|\{i: y_i = c\}|} \sum_{i:\, y_i = c} z_i$ the centroid of class c on $\mathbf{v}_1$. We define:

$$\mathrm{SNR} = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{within}}} = \frac{\mathrm{Var}_c\big[\, \bar{z}_c \,\big]}{\frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathrm{Var}\big[\, z_i \mid y_i = c \,\big]}. \tag{1}$$

The numerator captures how spread apart the class centroids are along the count axis; the denominator captures the average spread of samples within each class. A SNR $\gg 1$ indicates that count identity explains substantially more of the variance along $\mathbf{v}_1$ than within-class noise does, implying a well-defined, recoverable counting signal at that layer.

3.2 Quantifying the Unembedding Bottleneck

We formalize the unembedding-bottleneck hypothesis as the claim that the digit-token columns of W_U share no preferential alignment with the count directions identified by probing, and derive a direct, quantitative test of this claim. Let $P^* \in \mathbb{R}^{|\mathcal{C}| \times d}$ denote the weight matrix of a logistic regression probe trained on last layer residuals of a model. Each row of P^* defines the direction in residual space most discriminative of a particular count class. We compute the absolute cosine similarity between each probe direction and each digit-token column of W_U:

$$\mathrm{sim}(c, c') = \frac{|P^*_{c,:} \cdot W_U[:, \tau(c')]|}{\|P^*_{c,:}\| \, \|W_U[:, \tau(c')]\|}, \tag{2}$$

where $\tau(c)$ maps count c to its vocabulary index. Under the unembedding-bottleneck hypothesis, the diagonal entries in $\mathrm{sim}(c, c')$ should be no larger than off-diagonal entries.

3.3 Causal Validation via Probe-Guided LoRA

To causally validate the unembedding-bottleneck hypothesis, we test the prediction that a minimal correction at the diagnosed locus should restore counting by applying a rank-r LoRA [Hu et al., 2022] to W_U. If counting accuracy recovers, this confirms that the failure was a geometric misalignment.

Problem Formulation. Let $\mathcal{C} = \{0, 1, \ldots, 9\}$ denote the set of target counts and let $\tau \colon \mathcal{C} \to \{1, \ldots, V\}$ map each count to its corresponding token index in W_U. We restrict attention to the count-relevant columns of the unembedding matrix, $W_U^{(\mathcal{C})} = W_U[:, \{\tau(c)\}_{c \in \mathcal{C}}] \in \mathbb{R}^{d \times |\mathcal{C}|}$.

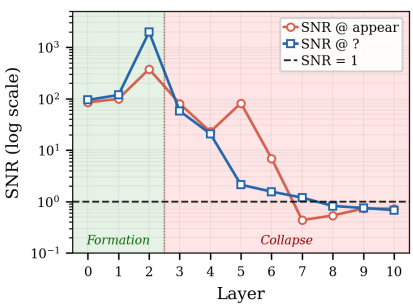

Figure 2: **Collapse of dominant-axis (PC1) SNR in Llama-2-13B.** The SNR along the dominant principal axis for both tokens of consideration peaks within the first few layers and then collapses below 1

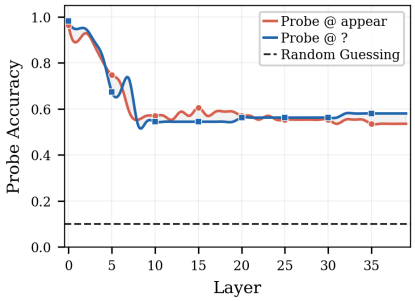

Figure 3: **Probe accuracy persists** Linear probes trained per layer recover the count with accuracy well above the random baseline across layers of Llama-2-13B, indicating that the count signal remains linearly decodable even in layers where the dominant-axis SNR is below 1.

Low-Rank Correction. We introduce the correction term $\Delta W = AB$ with $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times|\mathcal{C}|}$, $r \ll \min(d, |\mathcal{C}|)$ and $\mathbf{a}_i$ is the residual activation, then the corrected logits are:

$$\hat{\mathbf{z}}_i = \mathbf{a}_i^\top \left(W_U^{(\mathcal{C})} + A\,B\right) \in \mathbb{R}^{|\mathcal{C}|}. \tag{3}$$

Probe-Informed Initialization. A standard random initialization of A treats the correction as an unconstrained search over all d-dimensional directions, ignoring the diagnostic information already available from probing. Initializing A from the probe's leading singular vectors seeds the LoRA correction in the subspace that is known to carry the count signal, reducing the optimization to learning only the rotation and scaling (B) needed to project that subspace onto the digit-token columns. Concretely, let $P^* \in \mathbb{R}^{|\mathcal{C}|\times d}$ be the probe weight matrix trained on the residual stream activations $\{x_i\}_{i=1}^n$ and count labels $\{y_i\}_{i=1}^n$ with L_2 regularization. Then we define P^* as follows :

$$P^* = \arg\min_P \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\text{CE}}\big(P\,\mathbf{x}_i,\ y_i\big) + \frac{1}{2C}\|P\|_F^2. \tag{4}$$

Here $\mathcal{L}_{CE}$ is the standard cross entropy loss and we set coefficient $C = 0.1$. We compute the SVD of $P^{*\top} = U\,\Sigma\,V^\top$ and initialize $A = U_{:,\,1:r}$, thereby anchoring the column space of ΔW to the top-r directions of the probe-identified count subspace. B is initialized to zero so that $\Delta W = AB = \mathbf{0}$ at initialization, preserving the original model's behavior at the start of training. We optimize A and B by minimizing cross-entropy over training residuals

This design choice directly operationalizes our mechanistic diagnosis that as the bottleneck is a misalignment between the counting subspace and W_U. We compare this against randomly initialized variant in Appendix B to isolate the contribution of the probe-informed initialization.

4 Experiments

Models : We use various models from both architecture families, i.e MHA (Multi-Head Attention) and GQA (Group Query Attention) for our experiments. From the MHA family we use Llama-2-7B and Llama-2-13B. From GQA, we use Llama-3-8B, Qwen2-1.5B, Mistral-7B and Gemma-2-2B.

4.1 Experimental Setup and Datasets.

To rigorously evaluate the counting mechanisms of LLMs, we utilize multi-tiered datasets which range from canonical controlled prompts to standard benchmarks like MMLU [Hendrycks et al., 2020], ensuring that our findings are not artifacts of specific tokenizations or phrasing. We describe the experiments and the related datasets below. For further details please refer Appendix A

A. Canonical Probing for Mechanistic Analysis

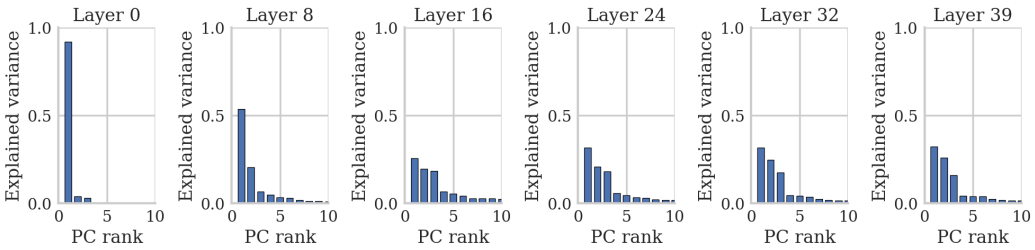

Figure 4: **Dispersion of count signal** Per-layer projections in Llama-2-13B show that count-discriminative variance is concentrated on PC_1 in early layers but spreads across many components in deeper layers, explaining the gap between the SNR collapse and the sustained probe accuracy.

To understand the lifecycle of the count signal, a set of 80 canonical prompts are designed for SNR analysis across model layers. Each prompt is of the format as shown in Figure 1. We test 10 distinct count values, i.e. the number of characters in the last line which are to be counted can lie in the range 0 to 9. And we have 8 prompt variants for each of the 10 count categories.

For the set of 80 prompts, we compute the SNR at layer ℓ as follows. We extract the residual stream activations $\{\mathbf{x}_i^{(\ell)}\}_{i=1}^{80}$ at the *appear* token yielding a matrix $X^{(\ell)} \in \mathbb{R}^{80\times d}$ where $\mathbf{x}_i^{(\ell)} \in \mathbb{R}^d$. We then follow procedure in Sec 3.1 to get the SNR. This dataset is also used to train linear probes at different layers of the model. In $X^{(\ell)} \in \mathbb{R}^{80\times d}$ each activation is paired with a count label $y_i \in \{0, 1, \ldots, 9\}$ and then we fit a logistic regression on this.

B. LoRA Hyperparameters

The dataset used for parametric repair and post-LoRA evaluations consists of 300 prompts which require the model to count a certain target entity which can be a letter,word or character. Counts of the target range from 0 to 9, with 30 deterministically seeded variants per count . Each prompt follows a 3-shot format as shown in Figure 1. We perform a rank=4 LoRA and use AdamW [Loshchilov and Hutter, 2017] with weight decay $= 10^{-4}$ as optimizer. Also, gradient clipping at 1.0 with a cosine-annealing learning-rate schedule with initial learning rate $= 0.01$ is used. Cross-entropy loss is used and training is done for 200 epochs. A 10-fold stratified cross-validation protocol over the 300 prompts is done to ensure that repairs generalize to held-out distributions without overfitting.

C. Evaluation Protocol

C.1 Compositional Generalization of the Counting Mechanism

We construct a combinatorial task pool for the counting problem by varying five factors: target category $c \in \mathcal{C} = \{\texttt{alphabetic}, \texttt{semantic}, \texttt{numeric}, \texttt{symbolic}, \texttt{nonce-word}\}$, token pair $(t_{\text{target}}, t_{\text{distractor}}) \sim \mathcal{T}_c$ drawn from a category-specific token set, target count $k \in \mathcal{K} = \{0, 1, \ldots, 9\}$ and prompt template $p \in \mathcal{P}$. The full task pool is the Cartesian product $\mathcal{S} = \mathcal{C} \times \mathcal{T}_c \times \mathcal{K} \times \mathcal{P}$. We uniformly sample without replacement a subset $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ and partition it into training and test sets with a 50/50 split, i.e. $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ respectively. Please refer Appendix A for more details. We evaluate the models generalization across the levels below :

- **Level 1 (Generalization across Seeds):** Training and test sets share the same combinatorial configurations but differ in random seed (i.e., the positional arrangement of tokens within a sequence). This tests whether the model memorizes exact input strings rather than learning the underlying counting function.
- **Level 2 (Generalization across Combinations):** Training and test sets contain disjoint configurations (c, t, k, p), i.e., $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$. This tests broader generalization beyond memorization of specific factor combinations. This setting simulates the common scenario that the specific combinations in test data are new, whereas the components are partially seen in training data.
- **Level 3 (Generalization across Token Pairs):** The token-pair sets are partitioned such that $\mathcal{T}_c^{\text{train}} \cap \mathcal{T}_c^{\text{test}} = \emptyset$ for each category c. In this most challenging setting, models must generalize to entirely unseen token identities, simulating the scenarios that exhaustive enumeration is infeasible.

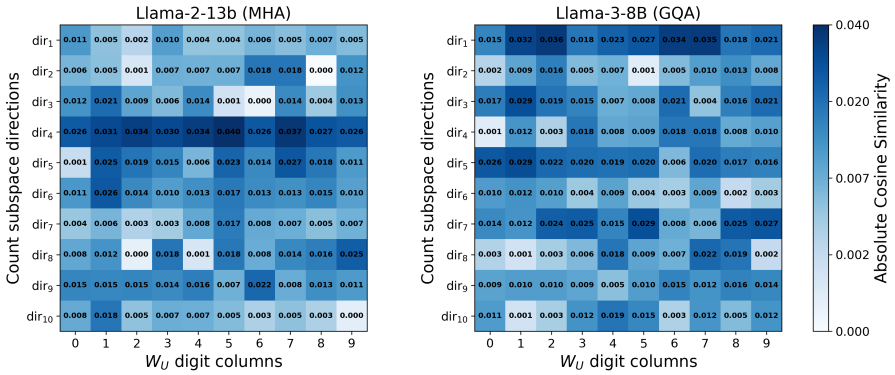

Figure 5: **Orthogonality of W_U to the count subspace.** Absolute cosine similarity between the 10 probe-derived count directions and the 10 digit-token columns of W_U in Llama-2-13B and Llama-3-8B. The diagonal values are no larger than off-diagonal ones

C.2 Retention of General Language Capabilities (RetGenCap):
To ensure that mechanistic surgery does not degrade broader capacities of the model, we use the models trained on D_{train} and evaluate them on the MMLU test set, spanning five diverse subjects: High School Mathematics, World History, Physics, Philosophy, and Virology. This tests whether targeted interventions preserve the model's general language understanding capabilities.

C.3 Extrapolation to Longer Sequence Lengths (ExLongSeq):
We stress-test the repair using sequences of length 20 i.e. $k \in \{0, \ldots, 19\}$. The three-shot exemplars in the prompt are adapted to the longer sequence length and the model is trained for 250 epochs. This tests if the LoRA repair generalizes to two-digit counting problems.

5 Results and Analysis

Before presenting our analysis of the research questions in Section 1, we define terms and conventions used. In the below sections L_N denotes the N-th model layer and PC_N denotes the N-th principal component. We refer to attention heads that cause a drop in the SNR as *killer heads*. These heads write large-norm vectors into the residual stream that are orthogonal to the count axis, inflating the total variance without geometrically erasing the count signal and thereby effectively burying it. First we explain how these heads are identified as they are important for the following analysis.

Identification of killer heads. For each head h in layer ℓ, we ablate its activations written into the residual stream at the *appear* token and recompute the SNR. A head whose removal raises the SNR dilutes the count signal when it is present. At L_5 of Llama-2-13B, head writes are nearly orthogonal to the count axis with a mean cosine similarity of 0.023 across 40 heads. Their effect therefore comes from adding variance off the count axis. Jointly removing the six strongest killer heads raises the SNR from 191.6 to 242.4 as shown in Table 1. Following this method, we identify the killer heads in models which serves as a basis for our mechanistic investigation.

Table 1: **Killer-head ablation of Llama-2-13B** The table shows the SNR and ΔSNR at the *appear* token after removing each head's write from the residual stream at Layer 5. Baseline SNR with all heads present is 191.6.

Removed head	H18	H14	H25	H23	H3	H4	All six
SNR	217.5	202.2	200.2	198.7	198.4	198.2	242.4
ΔSNR	+25.9	+10.6	+8.6	+7.1	+6.8	+6.6	+50.8

5.1 Analysis of RQ1

Lifecycle of count signal In this section, we explain the complete lifecycle of the count signal measured using SNR and linear probes. For this, we use Llama-2-13B and method discussed in Sec4.1(A). We find similar trends in other models too, but we use Llama-2-13B just an example

here. The prompts used to query the model are of fixed length and a sample is shown in Figure 1. In the causal structure of the prompt, *appear* is the token at which the computational task becomes fully specified i.e only after processing *"how many times does N appear"* does the model know it must count occurrences of a particular letter. Prior work [Wang et al., 2022] on transformer circuits shows that task-relevant information concentrates at the token that specifies the required computation so tracking SNR at *appear* lets us observe the count signal's behavior across model layers. We empirically observe that *appear* token and *?* token have the highest SNR after L_0 compared to other tokens and Figure 2 shows the variation in it across layers in Llama-2-13B. We observe that it peaks at initial layers and then dips and eventually flattens out in the end. We report SNR only for the initial layers (upto 10 here) in Figure 2, as values in deeper layers fall below 1, rendering them indistinguishable from noise.

Rotation of Count Subspace The reduction in SNR of the count signal manifests differently across attention architectures. In MHA models, a single dominant *killer head* accounts for most of the SNR collapse, whereas in GQA models the weight-sharing structure where each KV head serves multiple query heads distributes this destruction across many heads. In Llama-3-8B, 17 heads spanning L_3–L_4 jointly act as *killer heads*. The distributed destruction leads to continuous rotation of the count subspace as it propagates through the network. To diagnose this rotation, we train linear probes on each layer's residual stream and evaluate them zero-shot on every other layer, yielding a layer-by-layer transfer matrix (Figure 6). Two patterns emerge. First, the diagonal accuracies remain uniformly high in both architectures, confirming that count information is not erased at any layer and it remains linearly decodable when probed locally. Second, the off-diagonal accuracies differ between architectures: MHA models exhibit higher cross-layer transfer, indicating a relatively stable count subspace, while GQA models show a lower transfer indicating that probes trained at one layer fail to generalize to others. Taken together, these observations show that the count signal in GQA models survives in subspace that is reoriented more across layers as compared to MHA models.

Dispersion of Count Signal Figure 2 might suggest that the count signal is destroyed after the first few layers and eventually lost as SNR collapses to a value indistinguishable from noise. However, when we trained and evaluated linear probes on the activations of various layers of the model we find that the signal still exists. Figure 3 shows the accuracy of the linear probe for both tokens across all layers, which is well above a random guessing baseline, indicating that the count signal is not lost even when the SNR < 1. The next natural question is : if the signal persists, why does the SNR collapse? SNR measures variance along the first principal component of the residual stream. A high SNR requires the count-dependent variance to be concentrated along a single dominant direction; if the same total signal is spread across multiple directions, each individual component carries less variance relative to noise, and the SNR drops even though a linear probe, which can aggregate evidence across all directions, still recovers the count. Figure 4 confirms that the counting signal is dispersed across several principal component directions due to *killer heads* in deeper layers, diluting the projection onto the first principal component leading to decline in SNR.

Our analysis indicates that the count signal is progressively diluted, yet not eliminated, as it propagates through the model layers. This naturally raises the question: if the signal remains present and is linearly decodable, why does the model still fail at counting? We address this in the next section

5.2 Analysis of RQ2

We now investigate why language models fail at counting despite internally representing the required information. We first identify the point of failure in the computation and then propose a repair for the failure in single digit counting problems.

Diagnosing the Orthogonality of the Unembedding Matrix (W_U) Having established the methodology for quantifying the unembedding bottleneck in Section 3.2, we now report the results. We extract probe weight vectors at the final layer and compute the 10×10 absolute cosine similarity matrix between each probe direction and each digit-token column of W_U as shown in Figure 5 for Llama-2-13B and Llama-3-8B. If W_U could read the count signal, the diagonal entries (where probe class matches digit column) should be substantially larger than the off-diagonal entries. Instead, all entries are uniformly near zero and their magnitudes are indistinguishable from the expected cosine similarity between random unit vectors in $\mathbb{R}^d$ ($\approx \sqrt{2/(\pi d)} \approx 0.011$ for $d = 4096$). We observe the

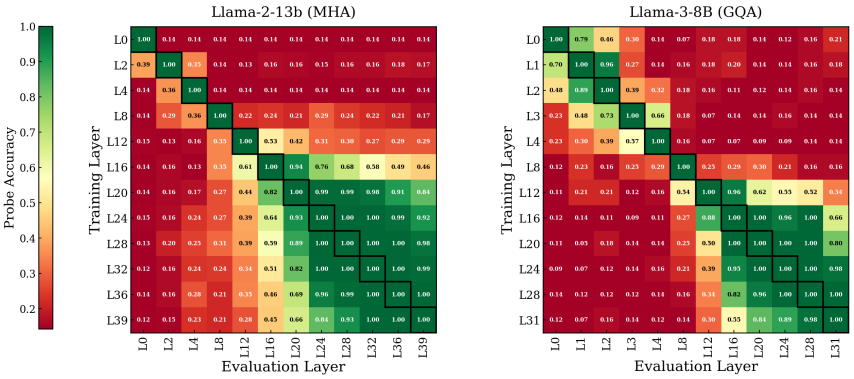

Figure 6: **Cross-layer probe transfer** Diagonal accuracies are uniformly high in both architectures, but off-diagonal transfer is lower in Llama-3-8B (GQA, mean 0.350) than in Llama-2-13B (MHA, mean 0.419), indicating more count subspace rotation in GQA models

same pattern across all six models. This is consistent with the unembedding bottleneck i.e. W_U's digit columns share no preferential alignment with the count subspace.

Aligning W_U via Probe-Informed LoRA. Having localized the failure to a geometric misalignment between the count subspace and the digit-token columns of W_U, we now turn to correcting it. Because the corrective update must inject the count subspace into W_U, the quality of the repair depends critically on how faithfully we can estimate that subspace from data. This is precisely where SNR becomes operationally useful i.e. layers with high SNR concentrate the count signal into a low-dimensional, high-quality subspace, yielding probe directions that are sharper and less contaminated by unrelated variance. We therefore use the per-layer SNR profile as a selection criterion for choosing the layer whose residual activations are used to train the probe P^* (Section 3.3). The leading singular vectors of $P^{*\top}$ then provide a principled initialization for the LoRA factor A, anchoring the column space of $\Delta W = AB$ to the probe-identified count subspace. In effect, SNR identifies where the count signal is cleanest and the probe identifies which directions carry it. The resulting correction modifies only $\sim 0.001\%$ of the model's parameters, providing a causal test of whether correcting W_U alone is sufficient to recover counting and the results of this are described in the next section.

Experimental Results Table 3 summarizes counting accuracy across all six models before and after probe-informed W_U LoRA under the evaluation settings of Section 4.1-C. All base models exhibit poor counting performance, with Level 1 accuracies ranging from 0 (Qwen2-1.5B) to 0.33 (Llama-3-8B) and Level 2 accuracies from 0.04 to 0.4, irrespective of model scale (1.5B-13B) or attention mechanism (MHA or GQA). LoRA fine-tuning yields large and consistent improvements across all models: in Level 1, absolute gains range from $+46.1\%$ (Llama-3-8B) to $+81.3\%$ (Qwen2-1.5B), while in Level 2 gains range from $+13.2\%$ (Llama-3-8B) to $+52.0\%$ (Llama-2-13B). Mistral-7B achieves the highest post-repair in both Level 1 and 2. Even the smallest model, Qwen2-1.5B, jumps from near-zero to 0.813 in Level 1, underscoring that targeted parametric repair is effective across diverse architectures and scales. In Level 3, we obtain an average accuracy gain of $\sim 25\%$ across models. The reduced headroom relative to Levels 1 and 2 is expected, as the LoRA must operate on token identities never seen during training. Even so, the magnitude of the improvement is substantial and consistent across models, providing additional empirical support for the W_U bottleneck. MMLU scores in the RetGenCap column remain virtually unchanged after fine-tuning for all models, confirming that the LoRA does not degrade broader language understanding.

Mechanism of the LoRA Repair We compute cosine similarity as described in 3.2 after the LoRA repair on Llama-2-13B, i.e. we use the corrected digit columns in place of the ones from the base model. We observe that the mean diagonal alignment between probe directions and their digit columns rises from 0.018 to 0.292 on average and every digit class improves its alignment with the respective count direction as shown in Table 2. The repair therefore works by re-orienting the digit columns toward the probe-identified count subspace, as the bottleneck hypothesis predicts, rather than through an unrelated parameter change.

Table 2: **Diagonal alignment in** $\mathrm{sim}(c, c')$ between probe directions and digit columns of W_U before and after probe-informed LoRA

Digit	0	1	2	3	4	5	6	7	8	9	Mean
Before	0.063	0.002	0.007	0.032	0.026	0.009	0.013	0.011	0.006	0.009	0.018
After	0.368	0.372	0.335	0.326	0.254	0.200	0.242	0.276	0.285	0.267	0.292

	Level 1		Level 2		RetGenCap		ExLongSeq	
Model	w/o LoRA	LoRA	w/o LoRA	LoRA	w/o LoRA	LoRA	w/o LoRA	LoRA
Llama-2-7B	0.011	0.725	0.095	0.532	0.372	0.371	0.070	0.138
Llama-2-13B	0.044	0.824	0.093	0.613	0.468	0.467	0.085	0.225
Mistral-7B	0.297	0.901	0.345	0.741	0.492	0.494	0.197	0.353
Llama-3-8B	0.330	0.791	0.407	0.539	0.533	0.534	0.168	0.305
Qwen2-1.5B	0.000	0.813	0.049	0.493	0.450	0.457	0.042	0.266
Gemma-2-2B	0.308	0.857	0.238	0.685	0.431	0.430	0.203	0.148

Table 3: **Performance of models pre and post W_U LoRA.** Across all six models, the LoRA repair yields gains on single-digit counting under both Level 1 and 2, while leaving MMLU performance (RetGenCap) essentially unchanged.

Preservation of digit generation. Apart from testing the preservation of performance of LoRA repaired models on generic token generation tasks which are covered under MMLU we also evaluate on digit token generation task like GSM8K Cobbe et al. [2021] dataset. Our analysis revealed that the LoRA learns to increase the scores of the digit-token columns whenever the residual stream contains a counting representation. However, the static correction has no mechanism to determine whether the current decoding position actually requires a digit. Consequently, when applied unconditionally throughout generation, it continually boosts digit logits even in places where the correct output is non-numeric, which leads to degradation in performance of the LoRA model on GSM8K. To address this issue, we use a *context-aware inference procedure* that introduces no additional trainable parameters. The procedure for it is as follows: (i) Compute the logits using the original unmodified W_U, (ii) If the top-1 prediction is not a digit keep the base model prediction unchanged. (iii) If the top-1 prediction is a digit, recompute only the 10 digit logits using the corrected W_U. (iv) After this single decision, generation proceeds entirely with the original unmodified W_U. With this setup the LoRA repaired Llama-2-13B performs at par with the base model on GSM8K dataset, both achieving an accuracy of 24%. This also reflects the mechanistic role of LoRA i.e. the base model retains the responsibility for deciding whether a digit is to be generated or not and the LoRA is used to improve which digit is selected when a numeric answer is already expected.

6 Limitation and Future Work

Table 3 - ExLongSeq column shows that LoRA finetuning is unable to restore the counting accuracy for multi-digit counting to the same extent as it does for single digit. We briefly discuss reasons for it in Appendix C and leave it for future work. We restrict our analysis to LLMs performing single-step generation and do not evaluate reasoning models that externalize intermediate counts via chain-of-thought (CoT) [Wei et al., 2022]. Also [Merrill and Sabharwal, 2023] show that CoT expands the class of problems transformers can solve relative to single forward pass but we intend to study problems completed in a single step. Also disentangling representation-level from procedure-level contributions in CoT models requires a separate analysis that we leave to future work.

7 Conclusion

In this work we present a mechanistic account of why LLMs fail at elementary counting despite possessing sufficient representational capacity. Through a framework combining SNR and linear probing, we traced the full lifecycle of the count signal across model layers and showed various characteristics of it. We show that the counting failure is not due to the absence of count information but due to a mismatch between a distributed internal representation and the model's existing vocabulary readout and propose a probe-informed LoRA that restores counting accuracy to validate out diagnosis. More broadly, our work studies a class of failures in LLMs where the information present in the model is inaccessible at the output which may extend to other scenarios beyond counting.

Acknowledgments

This work was supported in part by U.S. NSF grants DBI-2238093, DBI-2422619, IIS-2211597, and MCB-2205148. This work was also supported in part by an Amazon Research Award (Fall 2025 CFP).

References

Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8076–8084, 2019.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

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

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2(5):6, 2023b.

Liwei Che, Zhiyu Xue, Yihao Quan, Benlin Liu, Zeru Shi, Michelle Hurst, Jacob Feldman, Ruixiang Tang, Ranjay Krishna, and Vladimir Pavlovic. Counting circuits: Mechanistic interpretability of visual reasoning in large vision-language models. *arXiv preprint arXiv:2603.18523*, 2026.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240):1–113, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

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

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in neural information processing systems*, 36:70293–70332, 2023.

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, 2021.

Xuyang Guo, Zekai Huang, Zhenmei Shi, Zhao Song, and Jiahao Zhang. Your vision-language model can't even count to 20: Exposing the failures of vlms in compositional counting. *arXiv preprint arXiv:2510.04401*, 2025.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. *arXiv preprint arXiv:2601.04480*, 2026.

Tal Haklay, Hadas Orgad, David Bau, Aaron Mueller, and Yonatan Belinkov. Position-aware automatic circuit discovery. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2792–2817, 2025.

Hosein Hasani, Amirmohammad Izadi, Fatemeh Askari, Mobin Bagherian, Sadegh Mohammadian, Mohammad Izadi, and Mahdieh Soleymani Baghshah. Understanding counting mechanisms in large language and vision-language models. *arXiv preprint arXiv:2511.17699*, 2025.

Hosein Hasani, Mohammadali Banayeeanzade, Ali Nafisi, Sadegh Mohammadian, Fatemeh Askari, Mobin Bagherian, Amirmohammad Izadi, and Mahdieh Soleymani Baghshah. Mechanistic interpretability of large-scale counting in llms through a system-2 strategy. *arXiv preprint arXiv:2601.02989*, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Kuinan Hou, Marco Zorzi, and Alberto Testolin. Sequential enumeration in large language models. *arXiv preprint arXiv:2512.04727*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

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

Muhammad Fetrat Qharabagh, Mohammadreza Ghofrani, and Kimon Fountoulakis. Lvlm-count: Enhancing the counting ability of large vision-language models. *arXiv preprint arXiv:2412.00686*, 2024.

Bianca Raimondi and Maurizio Gabbrielli. Mechanistic interpretability of cognitive complexity in llms via linear probing using bloom's taxonomy. *arXiv preprint arXiv:2602.17229*, 2026.

Saurav Sengupta, Nazanin Moradinasab, Jiebei Liu, and Donald E Brown. Can vision-language models count? a synthetic benchmark and analysis of attention-based interventions. *arXiv preprint arXiv:2511.17722*, 2025.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, 2020.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

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

Gilad Yehudai, Haim Kaplan, Guy Dar, Royi Rassin, Asma Ghandeharioun, Mor Geva, and Amir Globerson. When can transformers count to n? *arXiv preprint arXiv:2407.15160*, 2024.

Xiang Zhang, Juntai Cao, and Chenyu You. Counting ability of large language models and impact of tokenization. *arXiv preprint arXiv:2410.19730*, 2024.

A Dataset and Compute Details

A.1 Compute Resources

We use 1 A100 80GB Nvidia GPU per run for all our training and evaluation experiments

A.2 Prompt Format and Construction

All prompts follow the 3-shot, single-line counting template introduced in Figure 1. A prompt consists of three fixed few-shot demonstrations followed by a single query line:

```
Given the sequence [demo1], how many times does {target} appear?  {a1}
Given the sequence [demo2], how many times does {target} appear?  {a2}
Given the sequence [demo3], how many times does {target} appear?  {a3}
Given the sequence [shuffled], how many times does {target} appear?
```

The first three lines are deterministic demonstrations and the fourth line is the test query: `[shuffled]` is a sequence of fixed length over a two-token alphabet $(t_{\text{target}}, t_{\text{distractor}})$ in which exactly k occurrences of t_{target} are placed at uniformly random positions, and the ground-truth label is k. Every prompt is generated under a deterministic seed of the form $\text{seed} = \alpha + k \cdot \beta + i$, where i indexes the variant and (α, β) are dataset-specific offsets.

A.3 Compositional Generalization Pool

For the evaluation protocol of Section 4.1 (C.1), we construct a combinatorial task pool over five factors, summarized in Table 4. Each draw from the pool yields a configuration (c, t, k, p) that fully specifies a single prompt up to the random shuffle seed.

B Comparison of LoRA Variants

To disentangle the role of the probe-informed initialization from the generic capacity of a low-rank correction at W_U, we compare two configurations under 10-fold stratified cross-validation. Both share the same rank, optimizer and training protocol described in Section 3.3; they differ *only* in how the A factor is initialized.

Factor	Values
Target category c	`alphabetic`, `semantic`, `numeric`, `symbolic`, `nonce-word`
Token pair $(t_{\text{target}}, t_{\text{distractor}}) \in \mathcal{T}_c$	`alphabetic`: A/B, X/Y, E/G; `semantic`: cat/dog, red/blue; `numeric`: 1/0, 7/3; `symbolic`: $*/-$, #/@; `nonce-word`: zzq/zzr, qzx/qzy
Target count k	$\{0, 1, \ldots, 9\}$
Prompt template $p \in \mathcal{P}$	p_0: *"Given the sequence {seq}, how many times does {target} appear?"* p_1: *"Sequence: {seq}. Count the number of {target} in the sequence:"* p_2: *"In the sequence {seq}, what is the total count of {target}?"* p_3: *"Given the sequence {seq}, how often does {target} occur?"*
Sequence length	9 tokens
Variants per configuration	30 (deterministically seeded)

Table 4: Factor decomposition of the compositional generalization pool $\mathcal{S} = \mathcal{C} \times \mathcal{T}_c \times \mathcal{K} \times \mathcal{P}$.

1. **Random-Init LoRA.** $A_0 \sim \mathcal{N}(0,\, 0.01^2\, I)$, $B_0 = \mathbf{0}$, with both A and B trainable. This variant tests whether a rank-r correction can discover the count subspace from scratch, with no mechanistic guidance from the probing analysis.
2. **Probe-Init LoRA (PI-LoRA).** $A_0 = U_{:,\,1:r}$ obtained from the SVD of the probe weight matrix P^*, with $B_0 = \mathbf{0}$ and both factors trainable. Optimization begins inside the probe-identified count subspace, so the dominant optimization burden shifts to learning the low-dimensional mapping B that rotates this subspace into the digit-token columns of W_U. A remains free to adapt, accommodating any residual mismatch between the probe subspace and the optimal correction direction.

Interpretation. The two variants differ in the prior they place on the failure locus. Random-Init makes no such assumption and must jointly discover the count subspace (A) and the readout mapping (B) through unconstrained optimization in $\mathbb{R}^{d\times r}$, which is prone to slower convergence and to settling in local minima that do not align with the true count subspace. PI-LoRA, in contrast, operationalizes the diagnostic finding of Section 5.1: by seeding A in the probe-identified subspace, it reduces the effective optimization problem to a low-dimensional rotation problem in B, while preserving the flexibility to refine A if needed. Under our central hypothesis that counting failures are caused by geometric misalignment at W_U, PI-LoRA should therefore converge to a higher-quality correction with the same training budget.

Empirical Comparison. Table 5 reports the post-training counting accuracy of both variants across the three generalization levels of Section 4.1 (C.1) on all models. Across every model and every generalization level, PI-LoRA matches or strictly improves over Random-Init. On Llama-3-8B, PI-LoRA improves Level 1 accuracy to 0.791 while on Mistral-7B Level 2 accuracy rises to 0.741. On Llama-2-7B and Llama-3-8B, Random-Init achieves only modest gains while PI-LoRA recovers between 13% and 34% across the three levels. On models where Random-Init is already strong (e.g., Qwen2-1.5B and Gemma-2-2B in Level 1), PI-LoRA still yields consistent, smaller improvements. Averaging over models and levels, PI-LoRA exceeds Random-Init on 14 of the 15 (model, level) cells, with the lone exception (Gemma-2-2B, Level 3) which is within 2.5%.

These results corroborate our central claim. The probe identifies the specific subspace in which the count signal lives, and seeding the LoRA correction in that subspace yields a uniformly better repair than the matched random-initialization baseline. The improvement is consistent across architectures and across model scales and most importantly across compositional generalization levels, including the token-pair held-out setting (Level 3) where the LoRA must operate on token identities never seen during training.

C Beyond Single Digits: Why W_U LoRA cannot resolve Multi-Token Counting Failure

The unembedding bottleneck described above explains failures for single-token counts, where correct prediction requires alignment between the residual stream and a single column of W_U. For larger counts, the model must instead emit a sequence of digit tokens, introducing additional constraints that

	Level 1		Level 2		Level 3	
Model	LoRA	PI-LoRA	LoRA	PI-LoRA	LoRA	PI LoRA
Llama-2-7B	0.549	0.725	0.400	0.532	0.320	0.371
Mistral-7B	0.670	0.901	0.528	0.741	0.415	0.493
Llama-3-8B	0.451	0.791	0.350	0.539	0.376	0.427
Qwen2-1.5B	0.725	0.813	0.488	0.493	0.367	0.381
Gemma-2-2B	0.813	0.857	0.597	0.685	0.435	0.410

Table 5: **Probe-Init LoRA outperforms Random-Init LoRA across models and generalization levels.** The table shows counting accuracy after rank-4 W_U LoRA fine-tuning under matched training budgets, comparing standard random initialization (LoRA) against probe-informed initialization (PI-LoRA). PI-LoRA improves over the random baseline in most cases confirming that seeding A in the probe-identified count subspace yields a uniformly better repair than unconstrained low-rank optimization.

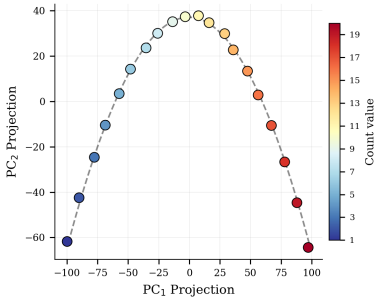

Figure 7: **Counts trace a smooth parabolic manifold in PC_1–PC_2.** Per-count means of the last layer residuals in Llama-2-13B for counts 1–20: PC_1 tracks magnitude almost linearly and adds quadratic curvature, showing that the model encodes count as a continuous quantity rather than as compositional digit features.

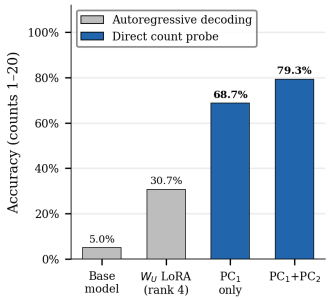

Figure 8: **Readout accuracy on counts 1-20** A probe using PC_1 alone reaches 68.7% accuracy and PC_{1-2} reaches 79%, while W_U LoRA at ranks 4–32 plateaus around 30%, indicating that the multi-digit failure is a representation–tokenization mismatch that cannot be resolved by W_U LoRA

are not captured by this single-step view. We show that failures in this regime arise from a mismatch between how the model internally represents count and how the tokenizer requires it to be expressed.

Geometry of the count representation. We compute PCA on 600 last layer residuals at the *appear* token for counts 1–20 (30 prompt variants per count) using Llama-2-13B, and plot the per-count mean of PC_1 vs. PC_2 in Figure 7. These two components capture 93% of the variance. Each count occupies a distinct location along a smooth, approximately parabolic curve. Quantitatively, PC_1 correlates strongly with count magnitude ($R^2 = 0.992$ under a linear fit), indicating that a dominant one-dimensional direction tracks numerical value. The curvature is captured by PC_2 ($R^2 = 0.985$ under a quadratic fit in k), suggesting that the representation lies on a low-dimensional, nonlinear manifold. These results show that the model encodes count as a continuous feature rather than as decomposed digit features.

Limits of W_U interventions. A linear probe using only PC_1 achieves 68.7% accuracy on counts 1–20, while a probe on PC_{1-2} reaches 79% (Figure 8), confirming that the residual stream contains information for count classification. However, this signal is not directly usable for generation. Probes map representations directly to 20 count classes, whereas the model must produce digit tokens autoregressively. For example, generating the count 12 requires first emitting `'1'` and then, conditioned on that output, emitting `'2'`. While autoregressive decoding allows the second step to depend on the first via the updated residual stream, modifying only W_U does not change how these intermediate representations are constructed. Instead, it can only reshape how existing representations are projected onto the vocabulary. This limitation is exacerbated by column sharing in W_U. The same output column $W_U[:, \tau(\texttt{'1'})]$ is used both for the single-digit count 1 and when it is the leading digit of multi-digit counts (10-19). Hence, gradient updates to this column must simultaneously support

conflicting roles, which cannot be cleanly separated without modifying upstream representations. Empirically, LoRA updates on W_U with ranks 4, 8, 16, and 32 all plateau at $\approx 30\%$ accuracy on counts 1–20. These results indicate that multi-token counting failures do not arise from a lack of information in the residual stream, but from a mismatch between representation and output structure. The model encodes count in a low-dimensional, continuous magnitude subspace, whereas the tokenizer requires a compositional, digit-level representation aligned with discrete vocabulary tokens. Interventions restricted to W_U cannot resolve this mismatch, as they do not alter how count information is structured within the residual stream.

D Dispersion of Count Signal

Generality of Count-Signal Dispersion Across Models. Figure 9 extends the per-layer principal-component analysis of Section 5.1 to all six models in our study. For each model, we project the residual-stream activations at the *appear* token onto the leading principal components at every layer and report the fraction of count-discriminative variance carried by each component. Across all architectures and scales, we observe the same qualitative pattern i.e in the early layers, count-discriminative variance is concentrated almost entirely on PC_1, producing the high SNR values that motivate our probe-layer selection. In intermediate and deeper layers, this variance progressively spreads across PC_2, PC_3, and beyond, while the dominant-axis SNR collapses below 1.This dispersion is consistent across both MHA models (Llama-2-7B, Llama-2-13B) and GQA models (Llama-3-8B, Mistral-7B, Qwen2-1.5B, Gemma-2-2B), and across model sizes spanning 1.5B-13B parameters. This shows that the dispersion phenomenon is not idiosyncratic to a particular architecture or attention mechanism but appears to be a general property of how transformer language models route count information through their residual stream. This generality is what makes the SNR-guided probe-layer selection in our LoRA pipeline a model-agnostic procedure i.e regardless of where the dispersion begins, the early layers provide a reliable layer at which the count subspace can be cleanly estimated.

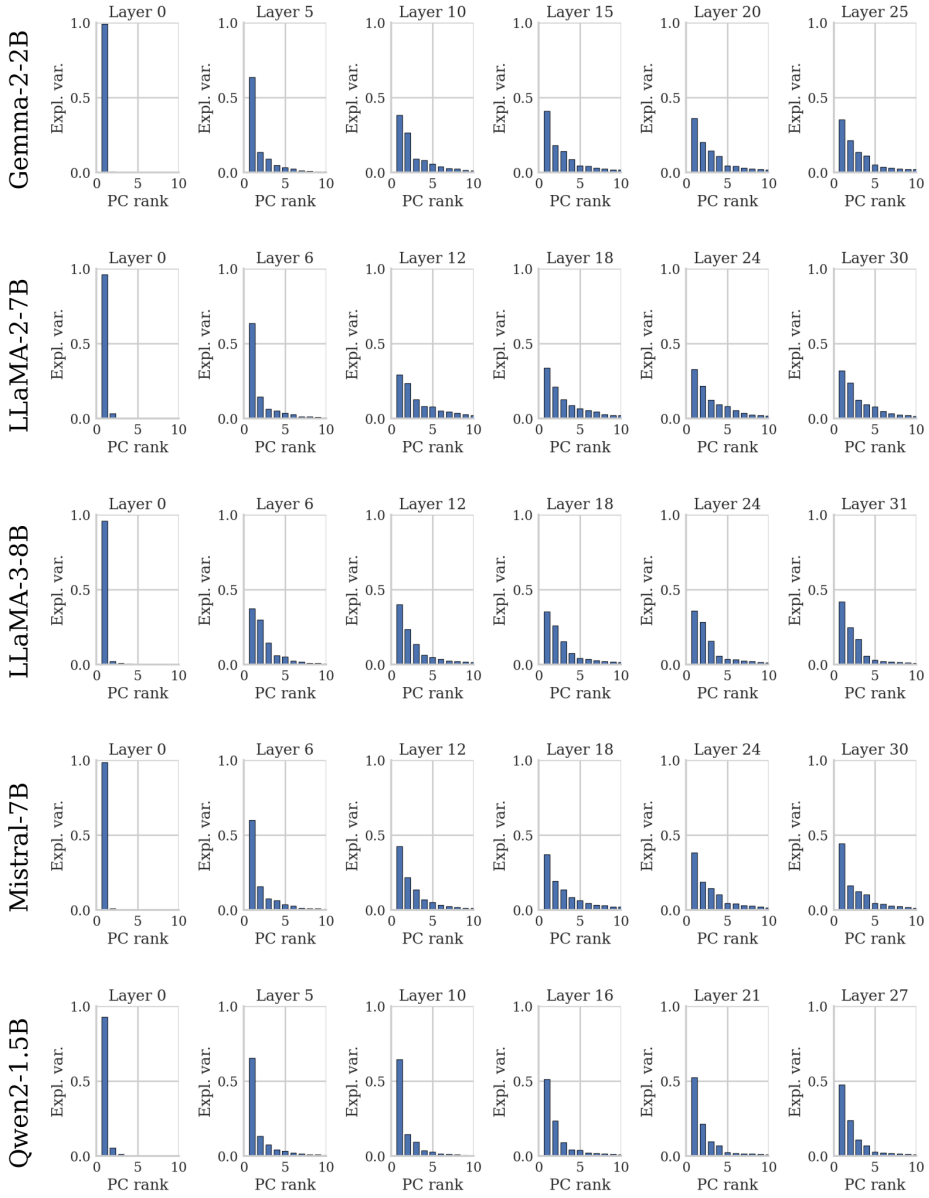

Figure 9: Dispersion Figures

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 5.1 and 5.2 provide experimental results for all the claims in introduction and abstract

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Section 6 has all the limitations listed

   Guidelines:

   - The answer [N/A] 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: [N/A]

   Justification: We do not provide an new formulas in the paper. All the formulas used are standard results like Signal to Noise Ratio and Cosine Similarity.

   Guidelines:

   - The answer [N/A] 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: Section 4.1 and Appendix A have all the details

   Guidelines:

   - The answer [N/A] 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: Code is attached with supplementary material

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 and Appendix A have all the details

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: The results reported in Table 3 is averaged over several models and data samples

   Guidelines:

   - The answer [N/A] 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 A has the information

   Guidelines:

   - The answer [N/A] 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: All the ethics have been followed

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: The work has no such societal impact

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work has no such risk

    Guidelines:

    - The answer [N/A] 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: All due diligence is followed

    Guidelines:

    - The answer [N/A] 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.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: All assets are documented

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: No crowdsourcing experiments in this work

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: Not applicable to this work

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: All the details are mentioned in the paper

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.