FINAL-LAYER JUMPS IN TRANSFORMER-LMS: CAUSES, A REMEDY AND INTERPRETABILITY EFFECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer language models (LMs) exhibit a final-layer jump: hidden states change gradually across the middle layers but shift abruptly at the final layer, a pattern widely observed in existing Transformer-LMs. The jump matters for Vocabulary Projection Methods such as the Logit Lens, which read out intermediate hidden states with the final LM head. Since the LM head is trained to decode the hidden state after the jump, a large final-layer change can reduce compatibility between intermediate hidden states and the LM head, degrading readouts at all intermediate layers. Through architectural ablations, we localize the jump to the final feed-forward network and show that it can be relocated: a linear buffer layer inserted before the LM head absorbs the jump, yielding a jump-relocated counterpart. Comparing models with and without the jump, we find that intermediate-layer representations remain highly similar while Logit Lens readouts change substantially, showing that the final-layer jump alters the results of Vocabulary Projection Methods and should therefore be taken into account.

1 INTRODUCTION

Transformer language models (Transformer-LMs) (Vaswani et al., 2017; Achiam et al., 2023; Grattafiori et al., 2024) have achieved strong performance across a wide range of tasks. It is important to understand their internal behavior for building more capable and safe models (Bereska & Gavves, 2024; Sun et al., 2024). Vocabulary Projection Methods (Rai et al., 2025) are a class of interpretability methods that project internal representations into the vocabulary space using the LM head to read out the information encoded in them (nostalgebraist, 2020; Katz & Belinkov, 2023; Sakarvadia et al., 2023; Pal et al., 2023; Elhelo & Geva, 2025). A representative example is the Logit Lens (nostalgebraist, 2020), which interprets hidden states in terms of vocabulary tokens.

Vocabulary Projection Methods rely on the premise that next-token predictions are progressively refined across Transformer layers. This refinement is reflected in gradual changes in hidden states across layers (Jiang et al., 2025). However, many Transformer-LMs exhibit an abrupt change in the hidden state at the final layer, which we refer to as the *final-layer jump* (Aubry et al., 2025; Gromov et al., 2025; Shibata et al., 2026). We quantitatively characterize the final-layer jump (Section 2.1) and show that it is widely observed across recent open-weight models (Figure 1). Because the LM head is trained to decode the hidden state after this final-layer jump, such a jump can reduce its compatibility with earlier hidden states, affecting vocabulary-space readouts at intermediate layers. This makes the final-layer jump a potential confound for Vocabulary Projection Methods.

In this study, we propose a simple architectural modification that shifts the final-layer jump outside the Transformer layers. Through ablation studies on training settings and model architectures, we empirically show that the pair of linear transformations closest to the output determines where the final-layer jump occurs (Section 3). Motivated by this finding, we insert a redundant linear layer, which we call a *buffer layer*, immediately before the LM head. The buffer layer and the LM head then form the pair of linear transformations closest to the output, shifting the jump from the final Transformer layer to this output-side linear pair (Section 4). After training, the buffer layer can be merged into the LM head, leaving the inference-time architecture unchanged.

Finally, we show that the final-layer jump affects intermediate-layer readouts obtained with Vocabulary Projection Methods. Using the Logit Lens, a representative Vocabulary Projection Method, we find

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

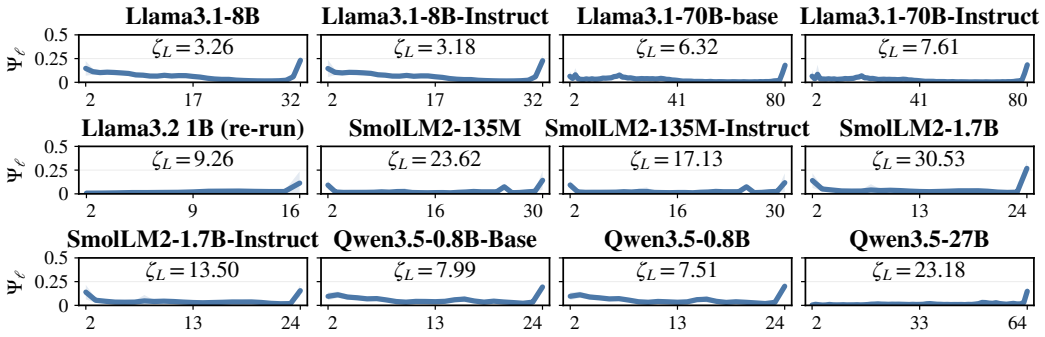


Figure 1: Layer-wise hidden-state displacement Ψ_ℓ and the final-layer jump score ζ_L (Equation (1)), averaged over the LAMBADA dataset. Here, Ψ_ℓ is the $[0, 1]$-normalized cosine distance between hidden states in consecutive layers. High ζ_L (> 3.0) indicates the scale of the displacement at the final layer is considered as an outlier.

that intermediate-layer readouts in the proposed model show higher agreement with the final prediction than those in the baseline model (Section 5). These results show that the outcomes of Vocabulary Projection Methods can differ depending on the presence or absence of the final-layer jump, which should therefore be accounted for in such analyses.

2 Hidden-State Trajectory

In this section, to quantitatively characterize the final-layer jump, a phenomenon in which hidden states change abruptly at the final layer, we introduce the hidden-state trajectory and the jump score, a metric for determining whether a hidden-state transition constitutes a jump. We then apply this metric to open-weight LMs and show that final-layer jumps are widely observed across multiple models.

2.1 Hidden-State Displacement

Let L be the total number of layers in Transformer-LMs. Hereafter, let $\boldsymbol{h}_\ell$ denote the output hidden-state vector of the ℓ-th layer, whose dimension is D, that is, $\boldsymbol{h}_\ell \in \mathbb{R}^D$, where $1 \leq \ell \leq L$ holds, and $\boldsymbol{h}_0$ represents an (input) embedding vector. We note that $\boldsymbol{h}_{\ell-1}$ is the input hidden-state vector of layer ℓ, and simultaneously the output hidden-state vector to layer $\ell - 1$. For a given token position, we represent the relationship between the input and output hidden-state vectors[1] as $\boldsymbol{h}_\ell = \mathtt{TFL}_\ell(\boldsymbol{h}_{\ell-1})$, where $\mathtt{TFL}_\ell(\cdot)$ denotes the ℓ-th Transformer layer, which primarily consists of an attention mechanism, a feed-forward network, and layer normalizations.

We interpret hidden-state transitions geometrically. From this perspective, each token prediction traces a trajectory in the hidden-state space that starts from $\boldsymbol{h}_1$ and terminates at the final-layer hidden-state vector $\boldsymbol{h}_L$. We characterize each step of the path by the angular change between consecutive hidden-state vectors, i.e., $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L)$.

Following prior work, we measure displacement based on cosine similarity (Tyukin et al., 2024; Li et al., 2025; Simoulin & Crabbé, 2021; Godey et al., 2024). Let $S_C(\boldsymbol{a}, \boldsymbol{b})$ be the cosine similarity between two arbitrary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ (See Appendix A.1).

Definition 1 (Hidden-State Displacement (Shibata et al., 2026)) *For a set of data points Ω, the displacement between the input and output hidden-state vectors at layer ℓ, denoted by $\Psi_\ell \in [0, 1]$, is defined as a normalized cosine distance and computed as follows:* $\Psi_\ell = \mathbb{E}_\Omega \left[\frac{1}{2}(1 - S_C(\boldsymbol{h}_{\ell-1}, \boldsymbol{h}_\ell))\right]$.

A larger Ψ_ℓ indicates a greater change in the normalized cosine distance between $\boldsymbol{h}_{\ell-1}$ and $\boldsymbol{h}_\ell$.

[1]For notational simplicity, we write the hidden-state at a single token position as $\boldsymbol{h}_\ell$, suppressing the token-position index and the dependence on the other tokens in the sequence.

Our aim is to identify outliers in hidden-state displacements, which we observe to occur primarily in the final layer of many Transformer-LMs. Here, we adopt the idea of standard outlier detection methods. One standard criterion that is robust to outliers is a robust z-score based on the median absolute deviation (MAD) (Iglewicz & Hoaglin, 1993). Let ζ_ℓ be the MAD-based robust z-score on the series of hidden-state displacements Ψ_ℓ with respect to $\ell \in \{2, \ldots, L\}$ that is defined as follows[2]:

$$\zeta_\ell = \frac{(\Psi_\ell - m)}{1.4826\, M}, \quad m = \texttt{median}_{j=2}^{L}(\Psi_j), \quad M = \texttt{median}_{j=2}^{L}(|\Psi_j - m|), \tag{1}$$

where $\texttt{median}_{j=2}^{L}(\Psi_j)$ denotes the median of $(\Psi_2, \ldots, \Psi_L)$.[3] We refer to the robust z-score ζ as *jump score* in this study.

Definition 2 (Outlier Hidden-State Displacement Detection) *Ψ_ℓ is flagged as a potential upper outlier when $\zeta_\ell > 3.0$. We define this event as a jump at layer ℓ. A final-layer jump means that $\zeta_L > 3.0$.*

The thresholding by 3.0 can be regarded as a robust analogue of the conventional three-sigma rule, in which the mean and standard deviation are replaced by the median and a MAD-based robust scale estimate, respectively. The use of the median absolute deviation for outlier detection is motivated by its robustness to extreme observations (Leys et al., 2013), and the resulting rule is consistent with the Hampel identifier and related Hampel filtering approaches based on median-centered, MAD-scaled deviations (Davies & Gather, 1993; Pearson et al., 2016).

2.2 Analysis of Open-Weight Models

We measured the displacement Ψ_ℓ across multiple open-weight models of various sizes with and without post-training to show that these models exhibit a pronounced hidden-state jump at the final layer. Specifically, we evaluated the Llama-3.1-8B, Llama-3.1-70B, Llama-3.2-1B (re-run)[4] (Grattafiori et al., 2024), SmolLM2-135M, 1.7B (Allal et al., 2025), Qwen3.5-0.8B and Qwen3.5-27B (Qwen Team, 2026). We use LAMBADA (Paperno et al., 2016) dataset as the inference dataset to measure the displacement of hidden-state vectors. LAMBADA is a dataset for predicting the final word given a long passage. In this study, we compute the average displacement across hidden-state vectors from all tokens in 5153 passages.

Figure 1 shows the displacement Ψ_ℓ and the final-layer jump score ζ_L (Equation (1)) for each model[5], demonstrating the presence of a final-layer jump across the evaluated models. Final-layer jumps are also observed on other inference datasets, including WinoGrande (Sakaguchi et al., 2021) and GSM8K (Cobbe et al., 2021) [6]. Our results show that final-layer jumps are widely observed across the evaluated open-weight Transformer-LMs. As described in Section 1, we consider relocating this jump outside the Transformer layers. Since this requires knowing where the jump arises, we next investigate which training and architectural factors are associated with it.

3 Ablation Study: Localizing the Final-Layer Jump

In Section 2.2, we quantitatively and visually confirm that a variety of open-weight models consistently exhibit the final-layer jump. This consistency suggests that the phenomenon is rooted in some fundamental aspect of standard Transformer-LMs methodology or design, as these models differ in training datasets, training procedures, architectural details, and model sizes. To identify where the jump is localized and which factors are associated with it, we conduct a comprehensive empirical ablation study, training a large number of models that differ in only one factor at a time. To the best of our knowledge, this is the first study to comprehensively investigate which factors relate to the behavior of the final-layer jump.

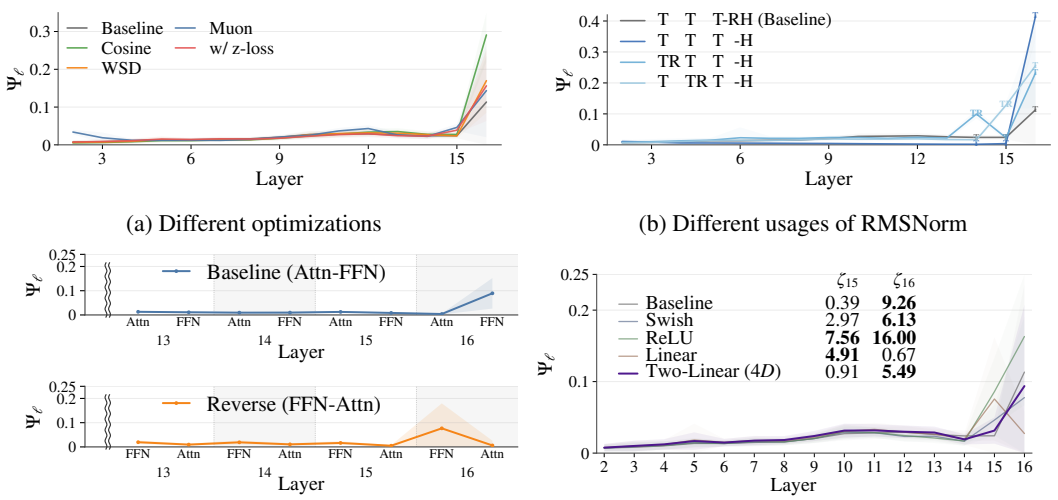
[2]We exclude Ψ_1 from the robust statistics because the transition from token embeddings to the first Transformer layer appears to be a different role from ordinary inter-layer transitions.

[3]In actual use, we treat $\zeta_\ell = 0$ when $M = 0$.

[4]Llama-3.2-1B (re-run) uses the same architecture as Llama-3.2-1B (Grattafiori et al., 2024), but is pre-trained by us. Details are provided in Section 3.1.

[5]Detailed jump score for other layers ℓ are shown in Appendix A.2.

[6]Details are discussed in Section A.4



(a) Different optimizations

(b) Different usages of RMSNorm

(c) Different orders of Attention and FFN

(d) Different FFN components

Figure 2: Displacement trajectory under different training and architectural settings. (a) corresponds to different LR scheduler, different optimizer, and z-loss settings. (b) corresponds to the different usages of RMSNorm, where T and R denote Transformer layer and RMSNorm, respectively. (c) corresponds to different orders of Attention and FFN. (d) corresponds to different FFN component variants. All models still exhibit that the jump happens at the near-final layers.

3.1 Baseline Setting

As a baseline, we pre-train a model with the same architecture as Llama-3.2-1B (Grattafiori et al., 2024), which consists of 16 layers with a hidden dimension of $D = 2048$ and vocabulary size $|\mathcal{V}| = 128256$. Let $\boldsymbol{x}$ be a token represented by a one-hot vector $\boldsymbol{x} \in \{0,1\}^{|\mathcal{V}|}$. $\boldsymbol{E} \in \mathbb{R}^{D\times|\mathcal{V}|}$ and $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}|\times D}$ denote the embedding matrix and the LM head matrix, respectively. As described in Section 2.1, $\texttt{TFL}_\ell(\cdot)$ represents the function of the ℓ-th Transformer layer, which takes as input $\boldsymbol{h}_{\ell-1}$ and returns $\boldsymbol{h}_\ell$ as:

$$\boldsymbol{h}_\ell = \texttt{TFL}_\ell(\boldsymbol{h}_{\ell-1}) = \boldsymbol{h}_\ell^{\texttt{Attn}} + \texttt{FFN}_\ell(\boldsymbol{h}_\ell^{\texttt{Attn}}), \text{ where } \boldsymbol{h}_\ell^{\texttt{Attn}} = \boldsymbol{h}_{\ell-1} + \texttt{Attention}_\ell(\boldsymbol{h}_{\ell-1}). \quad (2)$$

$\texttt{Attention}_\ell$ is implemented using pre-RMSNorm (Zhang & Sennrich, 2019) and grouped-query attention (GQA) (Ainslie et al., 2023), and $\texttt{FFN}_\ell$ adopts pre-RMSNorm and SwiGLU (Shazeer, 2020), which is expressed in terms of three parameter matrices $\boldsymbol{W}_\ell^{\texttt{Up}}, \boldsymbol{W}_\ell^{\texttt{Gt}} \in \mathbb{R}^{D_{\text{inter}}\times D}$, $\boldsymbol{W}_\ell^{\texttt{Dw}} \in \mathbb{R}^{D\times D_{\text{inter}}}$ and the Swish function (Prajit Ramachandran, 2018) as $\texttt{FFN}_\ell(\boldsymbol{h}_\ell^{\texttt{Attn}}) = \boldsymbol{W}_\ell^{\texttt{Dw}}(\texttt{Swish}(\boldsymbol{W}_\ell^{\texttt{Gt}}\boldsymbol{h}_\ell^{\texttt{FFNin}}) \odot \boldsymbol{W}_\ell^{\texttt{Up}}\boldsymbol{h}_\ell^{\texttt{FFNin}})$, where $\boldsymbol{h}_\ell^{\texttt{FFNin}} = \texttt{RMSNorm}(\boldsymbol{h}_\ell^{\texttt{Attn}})$. Here, $D_{\texttt{inter}}$ denotes the intermediate dimension of the FFN. Finally, the logit vector $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{V}|}$ is computed as follows:

$$\boldsymbol{z} = \boldsymbol{H}\,\texttt{RMSNorm}^{\texttt{out}}(\boldsymbol{h}_L), \quad (3)$$

where $\texttt{RMSNorm}^{\texttt{out}}(\cdot)$ denotes RMSNorm that is applied after the computation of the final transformer layer. Detailed model settings are provided in Appendix C. For training, we sample 200B tokens from the FineWeb-Edu (Penedo et al., 2024) corpus and train the model for 200K steps[7]. We optimize the model using AdamW (Loshchilov & Hutter, 2019) with the learning rate scheduler of Warmup-Stay-Only (WSO) (Yano et al., 2026). Detailed training settings are provided in Appendix D.

We show the displacement Ψ_ℓ of the baseline model in Figure 2a. The final-layer score also satisfies $\zeta_L = 9.26 > 3.00$, indicating that the baseline model exhibits a final-layer jump. Thus, we successfully reproduce the final-layer jump commonly observed in open-weight models, as discussed in Section 2.2. We use this model and training setting as the baseline hereafter.

[7]According to the Chinchilla scaling law (Hoffmann et al., 2022), the optimal training data size for a 1B model is approximately 20B tokens. In this study, we use 200B tokens (10× larger) for all models, which is comparable to the training setup of Pythia (Biderman et al., 2023).

3.2 Ablation Study

First, we train models with different learning rate schedulers (WSO, WSD, Cosine), optimizers (AdamW, Muon (Jordan et al., 2024)), regularizers (with or without Z-Loss (Chowdhery et al., 2023)), with or without the last RMSNorm, and changing the position where the RMSNorm applies ($\boldsymbol{h}_L$, $\boldsymbol{h}_{L-1}$, or $\boldsymbol{h}_{L-2}$). We briefly show the results in Figures 2a, 2b and provide the details in Appendix E. Despite the experiments covering various modifications of optimizations and architectures, we found that all the models still exhibit the final-layer jump. This result incentivizes us to conduct a fine-grained analysis of the jump in trajectories.

3.3 Analysis of FFN Component

In this section, we further localize the final-layer jump. By swapping the order of the attention and FFN, we find that the jump occurs at the FFN closest to the output (Figure 2c) [8]. We then conduct a more detailed analysis of which FFN components contribute to this phenomenon. We pre-train models with architectures in which the final-layer FFN is modified in four ways: replacing SwiGLU with Swish or ReLU (Nair & Hinton, 2010), removing the nonlinear activation function, and reducing a linear layer. These variants are summarized in Equation (4). The training and the other configurations are identical to those of the baseline.

$$\mathtt{FFN}_L(\boldsymbol{h}_L^{\mathtt{Attn}}) = f(\mathtt{RMSNorm}(\boldsymbol{h}_L^{\mathtt{Attn}})),\ f(\boldsymbol{h}) \in \left\{ \begin{matrix} \boldsymbol{W}^{\mathtt{Dw}}\mathtt{Swish}(\boldsymbol{W}^{\mathtt{Up}}\boldsymbol{h}),\ \boldsymbol{W}\boldsymbol{h}, \\ \boldsymbol{W}^{\mathtt{Dw}}\mathtt{ReLU}(\boldsymbol{W}^{\mathtt{Up}}\boldsymbol{h}),\ \boldsymbol{W}^{\mathtt{Dw}}\boldsymbol{W}^{\mathtt{Up}}\boldsymbol{h} \end{matrix} \right\}. \tag{4}$$

We present the displacement Ψ_ℓ for each model in Figure 2d. We also report the jump scores ζ_{L-1} and ζ_L ($L-1, L = 15, 16$). Note that no other layers have ζ_ℓ values greater than 3.0. For models in which the final-layer FFN consists of MLP or two linear matrices, the jump score ζ_L is 9.26, 6.13, 16.00 and 5.49, all of which exceed 3.0. In contrast, for the FFN composed of a single linear matrix ($\boldsymbol{W}\boldsymbol{h}$), $\zeta_L = 0.67 < 3.0$, indicating that no jump occurs in the final layer. Instead, the jump occurs in the preceding layer ($\zeta_{L-1} = 4.91 > 3.0$). Interestingly, the single-linear and the two-linear component result in different dynamics of the jump while they have the same function class.

4 Jump-Relocated Model

The previous section suggests that a final-layer jump mainly requires two linear transformations that are closest to the output. This observation suggests that the abrupt change need not be expressed entirely inside the final Transformer layer. Instead, if an additional linear transformation is placed immediately before the LM head, these two transformations become the ones closest to the output, and the jump is expected to be absorbed there rather than in the hidden-state trajectory. This section introduces a simple architectural modification for absorbing final-layer jumps into a redundant **buffer layer**: inserting a linear matrix immediately before the LM head.

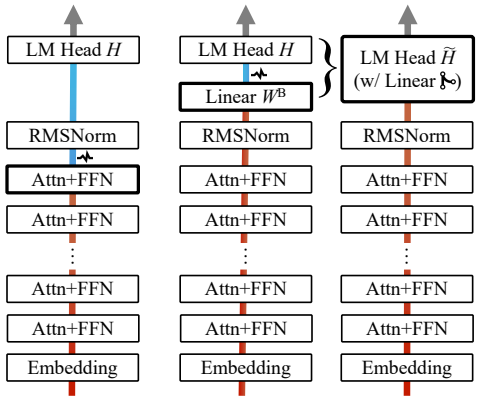


Figure 3: (a) The baseline shows the jump after the final Transformer layer. (b) The proposed model makes the inserted linear layer absorb the jump. The layer is merged into the LM head after training, giving an equivalent jump-relocated model.

4.1 Model Design

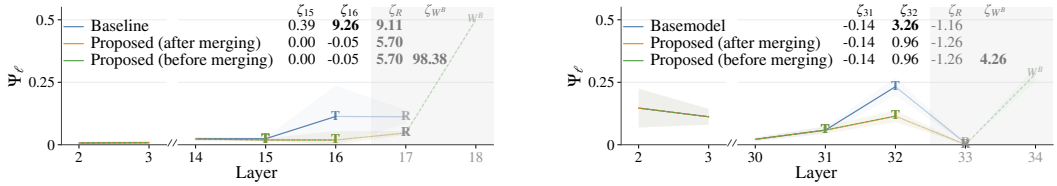
The proposed method in this study is simple: a buffer layer, which is a bias-free linear matrix $\boldsymbol{W}^{\mathtt{B}} \in \mathbb{R}^{D\times D}$, is inserted immediately before the LM head.

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{H}\boldsymbol{W}^{\mathtt{B}}\,\mathtt{RMSNorm}^{\mathtt{out}}(\boldsymbol{h}_L) \\ &= \widetilde{\boldsymbol{H}}\,\mathtt{RMSNorm}^{\mathtt{out}}(\boldsymbol{h}_L), \end{aligned} \tag{5}$$

[8] Detailed experimental settings and results are provided in Appendix E.6

Table 1: Validation loss and downstream performance of pre-trained models.

Method	Val. Loss ↓	Downstream Performance ↑							
		ARC-E	ARC-C	BoolQ	HellaSwag	OBQA	PIQA	Winogrande	**Average**
Baseline	2.494	68.6	33.1	59.4	41.8	28.0	70.8	56.0	51.1
Proposed	2.495	69.7	32.6	59.6	41.0	25.4	71.7	58.8	51.3



(a) Pre-trained models.

(b) Llama3.1-8B before and after additional training.

Figure 4: Displacement trajectories. The baseline models have the Transformer layers (T) with jumps, whereas the proposed models do not. Here, ζ_R and $\zeta_{\boldsymbol{W}^{\mathrm{B}}}$ correspond to the jump scores at the RMSNorm (R) and the buffer layer ($\boldsymbol{W}^{\mathrm{B}}$), respectively, which can be merged into the LM head.

where $\widetilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{W}^{\mathrm{B}}$. We can see later that this apparently redundant layer changes the behavior of optimization and the jump phenomenon. This modification introducing two-layer linear network increases the number of model parameters by $D \times D$ [9]. However, as shown in Equation (5) and Figure 3 (b), the added linear matrix $\boldsymbol{W}^{\mathrm{B}}$ can be absorbed into the adjacent linear LM head $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}| \times D}$ after training[10]. The computationally equivalent variant using the merged matrix $\widetilde{\boldsymbol{H}}$ has the same architecture as the baseline, removing any additional overheads during inference. This idea is inspired by post-training quantization methods (Park et al., 2025; Ashkboos et al., 2024a;b). All other model settings are kept the same as in the baseline (Section 3.1).

4.2 Training Settings

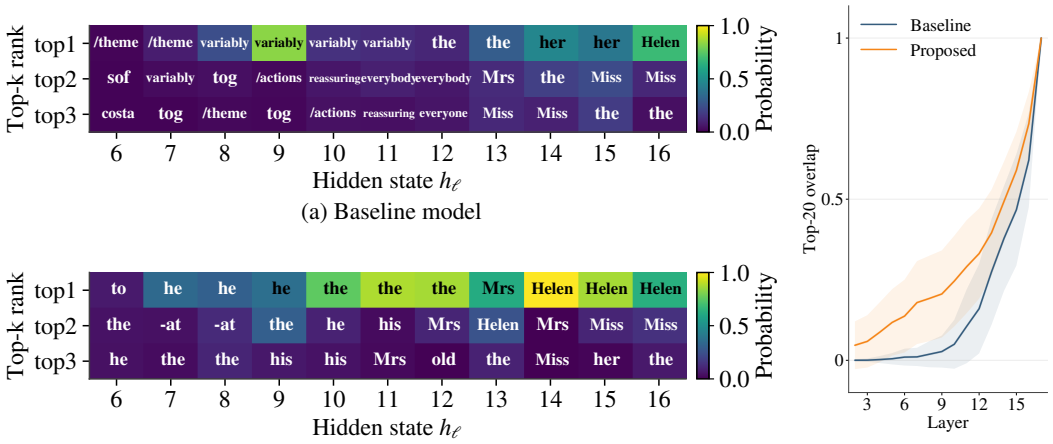
All pre-training settings are identical to those of the baseline model (Section 3.1). Details are provided in Appendix D.1. To investigate the impact on post-training performance, we perform supervised fine-tuning (SFT), after pre-training in this section. For the SFT dataset, we use tulu-v1-sft-mixture (Wang et al., 2023), which contains diverse instruction-following examples. Details are provided in Appendix D.2.

4.3 Sanity Check of Performance

First, to confirm that the modification does not harm the model performance, we employ several downstream tasks for evaluating the pre-trained models, measuring the accuracy of binary-choice QA on BoolQ (Clark et al., 2019) and multiple-choice QA on ARC-easy (ARC-e), ARC-Challenge (ARC-c) (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (OBQA) (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021). For the models after SFT, we use three benchmarks that evaluate instruction-following ability, MT-Bench (Zheng et al., 2023), Vicuna Bench (Zheng et al., 2023), and the WizardLM test set (Xu et al., 2024). We use GPT-4 as the judge model. The model performance of the pre-trained models is shown in Table 1. The average performance across tasks is comparable. This trend also holds after SFT. Details are provided in Appendix F.

[9] The total number of parameters in $\boldsymbol{W}^{\mathrm{B}}$ is 4.19M. The additional memory cost during training is approximately 67 MB.

[10] If the model uses *weight-tying*, we first need to *untie* it, that is, separate the embeddings from the LM head, and then merge the buffer weights into the LM head. This requires additional memory because the embeddings and LM head must be stored separately, but the underlying computational requirements remain identical to those of the original model.



(a) Baseline model

(b) Proposed model (jump-relocated model)

(c) Top-20 overlap

Figure 5: (a, b): Logit Lens visualizations for the next-token prediction, given the sentence ending with "You don't mean anything to me anymore,". Here, "Helen" (a person name) is the correct token. (c): Top-20 overlap (Equation (6)) across layers (shading: standard deviation). The jump-relocated model yields early-layer Logit Lens outputs more aligned with the final layer.

4.4 Hidden-State Displacement

As shown in Figure 4a, the baseline model family exhibits the final-layer jump, where the final Transformer-layer jump score ζ_{16} is $9.26 > 3$ for the pre-trained models. In contrast, in the proposed model (after and before merging the buffer layer), the corresponding scores are $\zeta_{16} = -0.05 < 3$, below the threshold. Post-trained models exhibit hidden-state trajectories similar to those of the pre-trained models[11]. We refer to the proposed model as a final-layer-jump-relocated model and **jump-relocated model** for short, as the final-layer jump is relocated from the final Transformer layer to the additional buffer layer between the Transformer stack and the LM head.

4.5 Applying Buffer Layer to Pre-trained Open-weight Models

Can the buffer layer be applied to already pre-trained models? To answer this question, we insert a buffer layer into pre-trained open-weight models and perform additional training, updating only the FFN in the final Transformer layer, $\texttt{RMSNorm}^{\texttt{out}}$, and the buffer layer while keeping all other parameters fixed[12]. Figure 4b shows the hidden-state displacement of Llama3.1-8B. The base model exhibits a final-layer jump with $\zeta_{32} = 3.26 > 3$, whereas after additional training the jump score decreases to $\zeta_{32} = 0.96 < 3$, indicating that the final-layer jump is absorbed into the buffer layer, while downstream task performance is largely preserved[13]. Thus, the buffer layer can be applied not only from the beginning of pre-training but also to already pre-trained models.

5 Analysis with Logit Lens

The Logit Lens interprets intermediate-layer predictions by projecting hidden states directly into the vocabulary space using the LM head. At each intermediate layer, we apply the model's output RMSNorm before projecting into the vocabulary: $\mathbf{z}_\ell^{\mathrm{LL}} = H\,\mathrm{RMSNorm}^{\mathrm{out}}(\boldsymbol{h}_\ell)$ for base model, and $\mathbf{z}_\ell^{\mathrm{LL}} = \widetilde{H}\,\mathrm{RMSNorm}^{\mathrm{out}}(\boldsymbol{h}_\ell)$, for proposed model. Here, $\widetilde{H}$ is the precomputed product of the LM head matrix and the buffer matrix. Each readout uses the parameters of the corresponding model. Since the LM head is trained to decode the final hidden state $\boldsymbol{h}_L$, applying it to intermediate states therefore assumes that $\boldsymbol{h}_\ell$ for $\ell < L$ is already readable by the same LM head. However, the

[11] Details are provided in Appendix E.8.

[12] Detailed experimental settings are described in Appendix J.1.

[13] Detailed downstream task performance is reported in Table 18.

final-layer jump makes this assumption questionable. A large transformation near the final layer can make the LM head suitable for decoding $\boldsymbol{h}_L$ but less suitable for decoding earlier hidden states.

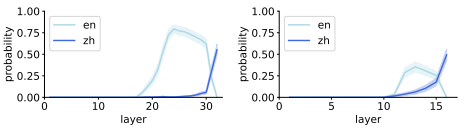
This section first compares the baseline model with the proposed jump-relocated model, whose representations evolve smoothly across layers. Both models are trained from scratch. We examine how the jump affects Logit Lens readouts both qualitatively and quantitatively. For the quantitative analysis, we use Top-K overlap.

Definition 3 (Top-K overlap) *For each model, let $S_\ell^{(K)}$ denote the set of top-K tokens at layer ℓ, and $S_L^{(K)}$ denote the set of top-K tokens at the final layer L. The Top-K overlap is defined as follows:*

$$\texttt{TopKOverlap}(\ell, K) = \frac{|S_\ell^{(K)} \cap S_L^{(K)}|}{K} \tag{6}$$

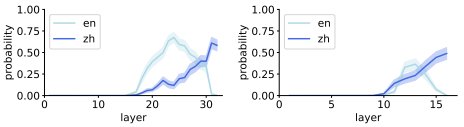
Top-K overlap measures how many of the model's final-layer prediction candidates already appear in the LM head readout at layer ℓ. This metric is not intended to measure correctness. We use it as a proxy for whether the LM head can meaningfully read the intermediate hidden state. A higher value indicates that the intermediate-layer representation is more compatible with the Logit Lens readout.

The results are shown in Figure 5. The jump-relocated model reveals candidates related to the final prediction at earlier layers than the baseline does (Figure 5(a,b)). Its higher mean Top-K overlap across multiple intermediate layers (Figure 5(c)) supports this qualitative pattern at the dataset level, even though the intermediate-layer representations of the two models remain highly similar (see Appendix K). We also observe higher Top-K overlap after applying the intervention in Section 4.5 to pre-trained open-weight models, with all layers before the final Transformer layer kept fixed (see Appendix J.3). These results caution against directly attributing late-emerging Logit Lens predictions to insufficient information or prediction difficulty in the baseline.



(a) Llama-3.1-8B, base model

(b) Llama-3.2-1B, base model



(c) Llama-3.1-8B, proposed

(d) Llama-3.2-1B, proposed

Figure 6: Comparison of multilingual behavior between the base models and the proposed models.

This concern also extends to multilingual processing. Prior work reports dominant English candidates at intermediate layers, with target-language candidates emerging later (Wendler et al., 2024). For non-English-to-Chinese word translation, we compare Logit Lens readouts of identical intermediate hidden-states before and after this intervention. During this intervention, the intermediate hidden states $\boldsymbol{h}_\ell$ for $\ell < L$ and the original LM head matrix H remain fixed, while the output RMSNorm and buffer are updated. Experimental details are described in Appendix J.4. Figure 6 shows the probabilities assigned to the correct Chinese and English tokens for the base models and the proposed models, in which a buffer layer is inserted followed by additional training. In the base models, consistent with the results reported in (Wendler et al., 2024), the probability of the correct English token increases in the intermediate layers. Near the final layers, the probability of the correct Chinese token increases and crosses over the probability of the English token. In contrast, in the models with a buffer layer, the probability of the Chinese token starts to increase from earlier layers than in the base models. This behavior substantially differs from the layer-wise behavior reported in (Wendler et al., 2024). These results show that the layer-wise behavior observed through Logit Lens depends substantially on the readout and cannot be equated with the point at which target-language information is formed. These empirical observations, along with the qualitative and quantitative results, show that readout effects can influence interpretations of prediction formation and language processing. Our method provides a way to assess this readout dependence and avoid overattributing readout behavior to the model's internal computation.

6 Related Work

Hidden-states have been widely used as a central object of analysis for understanding the internal behavior of Transformer-LMs. For example, prior studies have analyzed the role of Transformer

layers in pre-trained models based on the similarity or change of hidden-states across layers (Sun et al., 2025a; Lad et al., 2025), accelerated inference by exploiting redundant layers (Sun et al., 2025a; Men et al., 2025), and improved model architectures to make deeper layers more effective (Sun et al., 2025b; Team, 2026; Xie et al., 2025). In addition, hidden-state-based analysis methods, such as linear probing (Alain & Bengio, 2017; Liu et al., 2019; Hewitt & Liang, 2019; Belinkov, 2022), Logit Lens (nostalgebraist, 2020; Belrose et al., 2023), activation patching (Meng et al., 2022), and causal tracing (Zhang & Nanda, 2024), have been proposed and used to analyze internal representations and inference processes in language models. Thus, the behavior of hidden-states plays an important role in understanding, improving, and interpreting Transformer-LMs.

In this context, Shibata et al. (2026) studied layer-wise hidden-state changes and proposed a regularization method to suppress final-layer jumps, hypothesizing that they negatively affect model performance. However, their method requires regularizing multiple layers, making it difficult to isolate the effects of the presence and absence of the final-layer jump [14].

Moreover, Tuned Lens (Belrose et al., 2023) mitigates the mismatch between intermediate-layer representations and the final output distribution by additionally training layer-wise translators. However, Tuned Lens readouts depend on the translator's training data (He et al., 2026) and its objective of aligning intermediate representations with the final output, and therefore may not faithfully preserve signals present in the original model's intermediate representations (Wendler et al., 2024; Liioubashevski et al., 2025; Kargaran et al., 2025). Indeed, Wendler et al. (2024), who analyzed the internal reasoning language of multilingual models, used Logit Lens rather than Tuned Lens because such optimization could suppress English-like representations in intermediate layers. By absorbing the final-layer jump into a buffer layer that can be merged into the LM head, this study investigates how the presence or absence of the final-layer jump affects a family of interpretability methods that use the LM head as a readout for intermediate layers [15].

Limitations In this study, we experimentally investigated the factors underlying the final-layer jump, which is widely observed in Transformer-LMs. However, the experiments in Sections 4 and 5 are primarily based on Llama-3.2-1B. Since these experiments involve large-scale pre-training, it is not straightforward to comprehensively evaluate multiple architectures and model sizes. Therefore, whether the findings of this study also hold for other model families, architectures, and scales remains an important direction for future work. For example, our experiments in this study focus on dense models, and mixture-of-experts (MoE) architectures (Shazeer et al., 2017), which have been adopted in recent large-scale models, are outside the scope of this work. Additionally, it would be valuable to extend the investigation beyond Transformer-LMs to the Mamba architecture (Gu & Dao, 2024), which has gained increasing attention in recent years. Another important direction for future work is to theoretically derive the emergence of the final-layer jump from training dynamics.

7 Conclusion

We studied the final-layer jump of Transformer-LMs. Across open-weight models and controlled pre-training ablations, we found that the jump is robust to common training choices but is localized primarily to the final FFN, where a two-layer parameterization is a key empirical source. Motivated by this, we introduced a linear buffer before the LM head. This buffer layer absorbed the abrupt change in the hidden-state trajectory, and after training, it could be folded into the output matrix, yielding no inference-time overhead while maintaining comparable average downstream performance. This design also allowed us to compare models with and without a final-layer jump under comparable training, evaluation, and inference-time architectures. Finally, our Logit Lens analysis shows that the final-layer jump can substantially change intermediate-state readouts, and therefore should be accounted for when interpreting Transformer hidden-states. These findings caution against treating intermediate hidden-states as directly readable by the LM head when the model contains a large late-layer displacement.

[14] See Appendix I.1 for a detailed analysis.

[15] The effect of the final-layer jump on the Tuned Lens is discussed in Appendix I.2.

AI USE STATEMENT

We used generative AI tools at several stages of the study and manuscript preparation.

We consulted an LLM about implementation approaches for modifying the publicly available Meta Lingua codebase (Videau et al., 2024) to implement architectural ablations of the Transformer. For our analyses using Tuned Lens (Belrose et al., 2023) and multilingual analysis (Wendler et al., 2024), we also consulted an LLM about implementation approaches for modifying the code provided by the respective authors to support the analysis of models with an inserted buffer layer. As the authors are non-native English speakers, we used an LLM to help translate portions of drafts written in our native language into English. We also used generative AI tools to assist with creating and modifying figures, writing and editing code, searching for and summarizing the literature, identifying relevant work and locating bibliographic information for BibTeX entries, and editing the manuscript to improve readability.

Additionally, we used an LLM to check the internal consistency of the manuscript. This included feedback on how the experimental methodology was described and on whether the reported results supported the stated claims and interpretations. The authors assessed these suggestions and made all final scientific decisions.

We reviewed all AI-assisted work. We tested and reviewed the implementation to check that it behaved as intended, checked translated and edited text for accuracy, checked AI-assisted figures for consistency with the methods and reported results, and verified all references against their original sources. The authors take responsibility for the final content of this work, including all text, claims, code, and figures produced with the assistance of generative AI.

ETHICS STATEMENT

This work does not raise any specific ethical concerns.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architectures and training configurations in Appendix C, Appendix D and Appendix L to support reproducibility. The source code used for training, evaluation, and analysis is also included in the supplementary material.

REFERENCES

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, 2023.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2017. URL https://openreview.net/forum?id=ryF7rTqgl.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlícek, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big - data-centric training of a small language model. *CoRR*, abs/2502.02737, 2025. doi: 10.48550/ARXIV.2502.02737. URL https://doi.org/10.48550/arXiv.2502.02737.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=vXxardq6db.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240. Curran Associates, Inc., 2024b. doi: 10.52202/079017-3180. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b5b939436789f76f08b9d0da5e81af7c-Paper-Conference.pdf.

Murdock Aubry, Haoming Meng, Anton Sugolov, and Vardan Papyan. Transformer block coupling and its correlation with generalization in LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kvLenbZZgg.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 04 2022. ISSN 0891-2017. doi: 10.1162/coli_a_00422. URL https://doi.org/10.1162/coli_a_00422.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2023. URL https://arxiv.org/abs/2303.08112.

Leonard Bereska and Stratis Gavves. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ePUVetPKu6. Survey Certification, Expert Certification.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2397–2430. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick,

Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. URL http://jmlr.org/papers/v24/22-1144.html.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

P.L. Davies and U. Gather. The identification of multiple outliers. *Journal of the American Statistical Association*, 88(423):782–792, 1993. ISSN 0162-1459. doi: 10.2307/2290763.

Amit Elhelo and Mor Geva. Inferring functionality of attention heads from their parameters. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17701–17733, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.866. URL https://aclanthology.org/2025.acl-long.866/.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Nathan Godey, Éric Clergerie, and Benoît Sagot. Anisotropy is inherent to self-attention in transformers. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35–48, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.3. URL https://aclanthology.org/2024.eacl-long.3/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ngmEcEer8a.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling (COLM)*, 2024.

Matteo He, William F. Shen, Xinchi Qiu, and Nicholas D. Lane. Sparse readout prism: Explaining logit-lens scores in features instead of tokens, 2026. URL https://arxiv.org/abs/2609.01936.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2733–2743, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275/.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zhen Leng Thai, Kai Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies. *CoRR*, abs/2404.06395, 2024. doi: 10.48550/ARXIV.2404.06395. URL https://doi.org/10.48550/arXiv.2404.06395.

Boris Iglewicz and David C Hoaglin. *Volume 16: how to detect and handle outliers*. Quality Press, 1993.

Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. Tracing representation progression: Analyzing and enhancing layer-wise similarity. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vVxeFSR4fU.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Amir Hossein Kargaran, Yihong Liu, François Yvon, and Hinrich Schuetze. How programming concepts and neurons are shared in code language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26905–26917, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1379. URL https://aclanthology.org/2025.findings-acl.1379/.

Shahar Katz and Yonatan Belinkov. VISIT: Visualizing and interpreting the semantic information flow of transformers. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14094–14113, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.939. URL https://aclanthology.org/2023.findings-emnlp.939/.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference?, 2025. URL https://arxiv.org/abs/2406.19384.

Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. *JOURNAL OF EXPERIMENTAL SOCIAL PSYCHOLOGY*, 49(4):764–766, 2013. ISSN 0022-1031. URL http://doi.org/10.1016/j.jesp.2013.03.013.

Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-LN: Unleashing the power of deeper layers by combining pre-LN and post-LN. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BChpQU64RG.

Daria Lioubashevski, Tomer M. Schlank, Gabriel Stanovsky, and Ariel Goldstein. Looking beyond the top-1: Transformers determine top tokens in order. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2B11W1Z6ID.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1073–1094, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1112. URL https://aclanthology.org/N19-1112/.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1035. URL https://aclanthology.org/2025.findings-acl.1035/.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372. Curran Associates, Inc., 2022. doi: 10.52202/068431-1262. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf.

Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 807–814. Omnipress, 2010. URL https://icml.cc/Conferences/2010/papers/432.pdf.

nostalgebraist. interpreting gpt: the logit lens. *LessWrong*, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *CoRR*, abs/2501.00656, 2025. doi: 10.48550/ARXIV.2501.00656. URL https://doi.org/10.48550/arXiv.2501.00656.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In Jing Jiang, David Reitter, and Shumin Deng (eds.),

Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL), pp. 548–560, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.conll-1.37. URL https://aclanthology.org/2023.conll-1.37/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Jungwoo Park, Taewhoo Lee, Chanwoong Yoon, Hyeon Hwang, and Jaewoo Kang. Outlier-safe pre-training for robust 4-bit quantization of large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12582–12600, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.618. URL https://aclanthology.org/2025.acl-long.618/.

Ronald K. Pearson, Yrjö Neuvo, Jaakko Astola, and Moncef Gabbouj. Generalized hampel filters. *Eurasip Journal on Advances in Signal Processing*, 2016:1–18, December 2016. ISSN 1687-6172. doi: 10.1186/s13634-016-0383-6.

Guilherme Penedo, Hynek Kydl¥'¥i¥vcek, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. Fineweb2: One pipeline to scale them all – adapting pre-training data processing to every language, 2025. URL https://arxiv.org/abs/2506.20920.

Quoc V. Le Prajit Ramachandran, Barret Zoph. Searching for activation functions, 2018. URL https://openreview.net/forum?id=SkBYYyZRZ.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models, 2025. URL https://arxiv.org/abs/2407.02646.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Mansi Sakarvadia, Arham Khan, Aswathy Ajith, Daniel Grzenda, Nathaniel Hudson, André Bauer, Kyle Chard, and Ian Foster. Attention lens: A tool for mechanistically interpreting the attention head information retrieval mechanism, 2023. URL https://arxiv.org/abs/2310.16270.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Keigo Shibata, Kazuki Yano, Ryosuke Takahashi, Jaesung Lee, Wataru Ikeda, and Jun Suzuki. Suppressing final layer hidden state jumps in transformer pretraining. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 1236–1262, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.64. URL https://aclanthology.org/2026.findings-eacl.64/.

Antoine Simoulin and Benoit Crabbé. How many layers and why? An analysis of the model depth in transformers. In Jad Kabbara, Haitao Lin, Amandalynne Paullada, and Jannis Vamvas (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 221–228, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-srw.23. URL https://aclanthology.org/2021.acl-srw.23/.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=F7aAhfitX6.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 25219–25227. AAAI Press, 2025a. doi: 10.1609/AAAI.V39I24.34708. URL https://doi.org/10.1609/aaai.v39i24.34708.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 163104–163136. Curran Associates, Inc., 2025b. doi: 10.52202/085713-5446. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/eeb57fdf745eb31a3c7ef22c59a4661d-Paper-Conference.pdf.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis

Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Kimi Team. Attention residuals. *CoRR*, abs/2603.15031, 2026. doi: 10.48550/ARXIV.2603.15031. URL https://doi.org/10.48550/arXiv.2603.15031.

Georgy Tyukin, Gbetondji J-S Dovonon, Jean Kaddour, and Pasquale Minervini. Attention is all you need but you don't need all of it for inference of large language models, 2024. URL https://arxiv.org/abs/2407.15516.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ¥L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL https://github.com/facebookresearch/lingua.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah Smith, Iz Beltagy, and Hannaneh Hajishirzi. How far can camels go? exploring the state of instruction tuning on open resources. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 74764–74786. Curran Associates, Inc., 2023. doi: 10.52202/075280-3268. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ec6413875e4ab08d7bc4d8e225263398-Paper-Datasets_and_Benchmarks.pdf.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in English? on the latent language of multilingual transformers. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15366–15394, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.820. URL https://aclanthology.org/2024.acl-long.820/.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie E Everett, Alexander A Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=d8w0pmvXbZ.

Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. Residual: Transformer with dual residual connections, 2023. URL https://arxiv.org/abs/2304.14802.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mhc: Manifold-constrained hyper-connections. *CoRR*, abs/2512.24880, 2025. doi: 10.48550/ARXIV.2512.24880. URL https://doi.org/10.48550/arXiv.2512.24880.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CfXh93NDgH.

Kazuki Yano, Shun Kiyono, Sosuke Kobayashi, Sho Takase, and Jun Suzuki. Pre-training LLM without learning rate decay enhances supervised fine-tuning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JnebU2QLdH.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d¥textquotesingle Alch¥'e-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Hf17y6u9BC.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

A Additional Analysis of open-weight model displacement

A.1 Cosine Similarity

Let $S_C(\boldsymbol{a}, \boldsymbol{b})$ be the cosine similarity between two arbitrary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, expressed as:

$$S_C(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}^\top \boldsymbol{b}}{\|\boldsymbol{a}\|_2 \, \|\boldsymbol{b}\|_2}, \tag{7}$$

where $\| \cdot \|_2$ represents the Euclidean norm of a given vector.

A.2 DETAILS OF FIGURE 1 RESULTS

Table 2: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models, computed using the cosine-distance metric.

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Llama3.1-8B	32	**3.26**	-0.14	-0.85	-0.95	–
Llama3.1-8B-Instruct	32	**3.18**	-0.10	-0.81	-0.91	–
Llama3.1-70B	80	**6.32**	2.85	-0.17	-0.34	–
Llama3.1-70B-Instruct	80	**7.61**	**3.34**	-0.08	-0.23	–
Llama3.2-1B (re-run)	16	**9.26**	0.39	0.38	0.57	–
Llama3.2-1B-Instruct (re-run)	16	**10.28**	0.45	0.42	0.61	–
Llama3.2-1B	16	2.88	-1.25	-1.49	-1.44	–
Llama3.2-1B-Instruct	16	2.48	-1.14	-1.41	-1.36	–
SmolLM2-135M	30	**23.62**	1.52	0.57	-0.67	{2, 25}
SmolLM2-135M-Instruct	30	**17.13**	1.07	0.46	-0.67	{2, 25}
SmolLM2-1.7B	24	**30.53**	-1.69	-2.01	-1.54	2
SmolLM2-1.7B-Instruct	24	**13.50**	-1.80	-1.97	-1.49	2
SmolLM3-3B-Base	36	2.28	1.22	-0.79	-1.36	–
SmolLM3-3B	36	1.59	0.67	-1.07	-1.61	–
Qwen3.5-0.8B-Base	24	**7.99**	-0.57	-1.19	-0.80	3
Qwen3.5-0.8B	24	**7.51**	-0.46	-1.04	-0.72	3
Qwen3.5-27B	64	**23.18**	2.85	0.75	0.75	{51, 52}
Gemma4-E2B	35	0.71	-0.29	-0.96	-1.25	–
Gemma4-31B	60	**28.12**	**17.44**	**13.29**	**3.98**	{2, 53, 54, 56}

Table 2 shows the jump score for layers L-3 through L, as well as for other layers with a jump score greater than 3. In Figure 7, we show the displacement trajectories and final-layer jump score ζ_L for Gemma4-E2B, Gemma4-31B (Team et al., 2026), and SmolLM3 (Bakouch et al., 2025). The evaluation protocol is the same as that described in Section 2.2. For both SmolLM3 and Gemma4-E2B, many intermediate layers exhibit large displacement. As a result, these layers are not identified as outliers, and their jump scores remain below 3.

Unlike our re-run, the original Llama-3.2-1B was obtained by structured pruning from Llama-3.1-8B followed by knowledge distillation, rather than standard pre-training from scratch (Meta, 2024). This difference in training procedure may contribute to the different hidden-state trajectories.

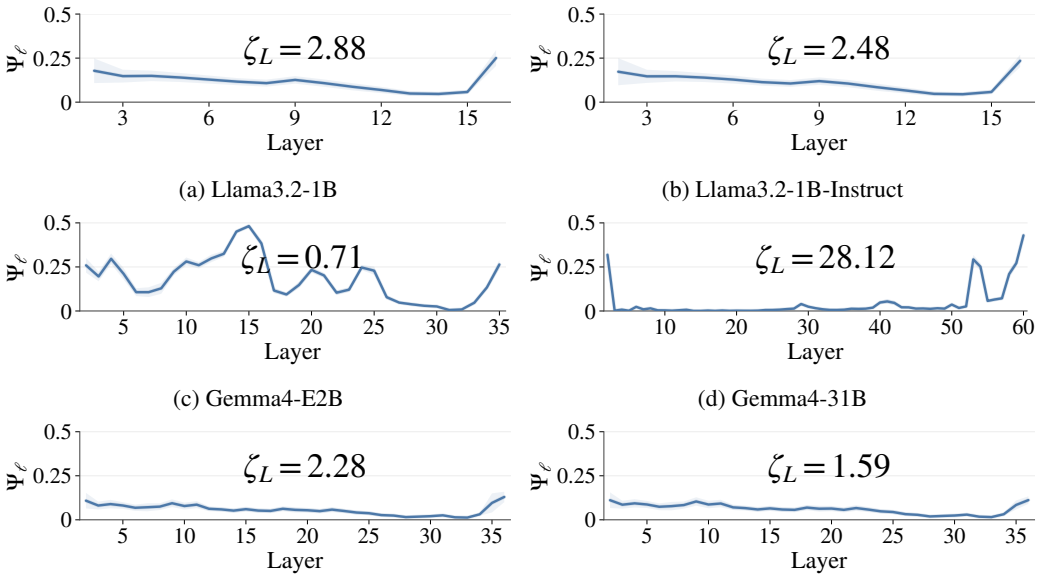
A.3 Displacement on Other definition of displacement

In the main text, we use cosine distance as the metric for hidden-state displacement. In this section, we define displacement based on the L2 distance and examine whether the final-layer jump can also be observed using different displacement metrics. Specifically, we define the absolute L2 displacement and relative L2 displacement as follows.

$$\Psi_\ell^{\mathtt{L2}} = \mathbb{E}_\Omega \left[||\boldsymbol{h}_\ell - \boldsymbol{h}_{\ell-1}||_2 \right] \tag{8}$$

$$\Psi_\ell^{\mathtt{rel-L2}} = \mathbb{E}_\Omega \left[\frac{||\boldsymbol{h}_\ell - \boldsymbol{h}_{\ell-1}||_2}{||\boldsymbol{h}_{\ell-1}||_2} \right] \tag{9}$$

$\Psi_\ell^{\mathtt{L2}}$ (Equation (8)) measures the L2 distance between the input and output hidden-states of layer ℓ. However, the norm of hidden-states tends to increase approximately linearly with layer



(a) Llama3.2-1B (b) Llama3.2-1B-Instruct

(c) Gemma4-E2B (d) Gemma4-31B

(e) SmolLM3-3B-Base (f) SmolLM3-3B

Figure 7: Layer-wise hidden state displacement Ψ_ℓ for next-word prediction on 5153 tokens from the LAMBADA dataset.

Table 3: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models, computed using L2 distance (Equation (8))

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Llama3.1-8B	32	**17.55**	**5.14**	2.09	1.37	–
Llama3.1-8B-Instruct	32	**18.41**	**5.41**	2.23	1.43	–
Llama3.1-70B	80	**65.70**	**39.70**	**11.04**	**8.45**	{4, 74, 75, 76}
Llama3.1-70B-Instruct	80	**85.54**	**44.62**	**13.07**	**10.44**	{4, 73, 74, 75, 76}
Llama3.2-1B (re-run)	16	**9.82**	**3.83**	2.61	2.03	–
Llama3.2-1B-Instruct (re-run)	16	**9.28**	**3.61**	2.39	1.86	–
Llama3.2-1B	16	**13.85**	2.32	1.10	0.59	2
Llama3.2-1B-Instruct	16	**10.52**	2.08	0.90	0.45	–
SmolLM2-135M	30	**20.57**	**12.94**	**5.73**	2.97	{12, 25}
SmolLM2-135M-Instruct	30	**19.94**	**13.20**	**6.09**	**3.22**	{12, 25}
SmolLM2-1.7B	24	**29.41**	**5.93**	**3.71**	**3.23**	{8, 20}
SmolLM2-1.7B-Instruct	24	**22.52**	**5.74**	**3.59**	**3.20**	{8, 20}
SmolLM3-3B-Base	36	**27.06**	**14.15**	**8.26**	1.59	2
SmolLM3-3B	36	**22.61**	**13.45**	**6.84**	1.68	2
Qwen3.5-0.8B-Base	24	**19.74**	**5.29**	**3.55**	**3.72**	{19, 20}
Qwen3.5-0.8B	24	**17.71**	**4.67**	2.92	**3.20**	–
Qwen3.5-27B	64	**36.57**	**11.52**	**7.09**	**6.69**	{51, 52, 53, 54, 55, 56, 57, 58, 59, 60}
Gemma4-E2B	35	0.83	0.34	-0.70	-1.52	–
Gemma4-31B	60	0.93	1.00	0.82	0.00	53

depth (Sun et al., 2025b; Xie et al., 2023). Therefore, directly comparing the displacement across layers using the absolute L2 displacement may be affected by this layer-dependent scale.

$\Psi_\ell^{\mathtt{rel-L2}}$ (Equation (9)) normalizes the L2 displacement by the norm of the input hidden-state to layer ℓ. This allows us to measure the magnitude of the residual update relative to the scale of the input hidden-state at each layer. The jump score ζ_ℓ is defined in the same manner as in Equation (1).

Tables 3, 4 show the jump scores computed using $\Psi_\ell^{\mathtt{L2}}$ and $\Psi_\ell^{\mathtt{rel-L2}}$, respectively. When using the absolute L2 displacement $\Psi_\ell^{\mathtt{L2}}$, layers satisfying $\zeta_\ell > 3$ tend to be concentrated in the later layers. This tendency may partially reflect the increase in the scale of hidden-states with layer depth. In

Table 4: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models, computed using relative L2 distance Equation (9)

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Llama3.1-8B	32	**3.18**	0.00	-0.95	-1.11	2
Llama3.1-8B-Instruct	32	2.92	-0.01	-0.84	-1.00	2
Llama3.1-70B	80	**4.23**	2.04	-0.10	-0.24	4
Llama3.1-70B-Instruct	80	**4.71**	2.20	-0.02	-0.13	4
Llama3.2-1B (re-run)	16	**3.69**	0.18	0.25	0.82	–
Llama3.2-1B-Instruct (re-run)	16	**3.39**	0.07	0.10	0.67	–
Llama3.2-1B	16	**3.12**	-0.97	-1.33	-1.14	2
Llama3.2-1B-Instruct	16	**3.31**	-1.17	-1.61	-1.46	2
SmolLM2-135M	30	**7.11**	0.75	1.07	-0.65	{2, 3, 12, 25}
SmolLM2-135M-Instruct	30	**6.53**	0.67	1.05	-0.67	{2, 12, 25}
SmolLM2-1.7B	24	**9.59**	-0.89	-0.67	-0.32	{2, 8}
SmolLM2-1.7B-Instruct	24	**7.40**	-1.14	-1.00	-0.52	{2, 8}
SmolLM3-3B-Base	36	**5.62**	1.64	-0.82	-1.92	2
SmolLM3-3B	36	**4.76**	1.45	-1.18	-2.26	2
Qwen3.5-0.8B-Base	24	**5.39**	-0.76	-1.24	-0.59	–
Qwen3.5-0.8B	24	**4.60**	-0.76	-1.27	-0.67	–
Qwen3.5-27B	64	**16.41**	**3.45**	1.07	1.45	{4, 51, 52, 55}
Gemma4-E2B	35	0.15	-0.34	-1.15	-1.75	–
Gemma4-31B	60	**6.18**	**6.84**	**5.09**	2.33	{2, 53, 54}

contrast, when using the relative L2 displacement $\Psi_\ell^{\texttt{rel}-\texttt{L2}}$, a particularly large displacement is still observed at the final layer even after normalization by the norm of the input hidden-state.

These results suggest that the final-layer jump is not specific to cosine-based displacement, but is also observed as a large residual update relative to the scale of the input hidden-state.

A.4 Displacement on Other datasets

Table 5: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models evaluated on GSM8K, computed using the cosine-distance metric.

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high-ζ layers
Llama3.1-8B	32	**3.38**	-0.62	-1.30	-1.48	–
Llama3.1-8B-Instruct	32	2.91	-0.66	-1.20	-1.35	–
Llama3.1-70B	80	**5.89**	2.58	0.07	-0.13	–
Llama3.1-70B-Instruct	80	**4.26**	1.88	-0.09	-0.23	–
Llama3.2-1B	16	**5.74**	-3.20	-3.82	-3.85	–
Llama3.2-1B-Instruct	16	**4.09**	-2.49	-2.99	-2.96	–
SmolLM2-135M	30	**18.51**	0.20	-0.04	-0.93	{2, 21, 25}
SmolLM2-135M-Instruct	30	**20.07**	0.34	0.18	-0.81	{2, 21, 24, 25}
SmolLM2-1.7B	24	**11.73**	-1.83	-1.68	-1.28	2
SmolLM2-1.7B-Instruct	24	**7.12**	-1.20	-0.93	-0.37	2
SmolLM3-3B-Base	36	2.77	1.46	-1.16	-1.74	–
SmolLM3-3B	36	1.94	0.93	-0.95	-1.35	–
Qwen3.5-0.8B-Base	24	**5.00**	-0.61	-1.44	-1.30	–
Qwen3.5-0.8B	24	**4.54**	-0.38	-1.19	-1.09	–
Qwen3.5-27B	64	**10.86**	2.82	0.76	0.79	{51, 52}
Gemma4-E2B	35	0.66	-0.31	-0.93	-1.12	–
Gemma4-31B	60	**28.59**	**17.34**	**9.42**	**3.87**	{2, 41, 53, 54, 55, 56}

In this section, we show that the final-layer jump is not specific to the distribution of the dataset used for measuring hidden-state displacement by evaluating it on different datasets. In the main text, we use the LAMBADA dataset, which requires predicting the final word of a passage, to measure hidden-state displacement. Here, we additionally use GSM8K (Cobbe et al., 2021), which requires multi-step mathematical reasoning, and WinoGrande (Sakaguchi et al., 2021), which evaluates commonsense reasoning ability, and measure the jump score in the same manner.

Table 6: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models evaluated on GSM8K, computed using relative L2 distance.

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high-ζ layers
Llama3.1-8B	32	2.95	-0.24	-1.02	-1.19	–
Llama3.1-8B-Instruct	32	2.40	-0.35	-1.03	-1.16	–
Llama3.1-70B	80	**4.68**	2.15	0.12	-0.08	–
Llama3.1-70B-Instruct	80	**3.41**	1.82	-0.03	-0.18	–
Llama3.2-1B	16	**3.04**	-0.94	-1.36	-1.41	–
Llama3.2-1B-Instruct	16	**3.10**	-0.98	-1.47	-1.46	–
SmolLM2-135M	30	**7.11**	0.00	0.43	-1.10	{2, 25}
SmolLM2-135M-Instruct	30	**7.30**	0.10	0.56	-0.96	{2, 25}
SmolLM2-1.7B	24	**6.55**	-0.77	-0.92	-0.66	2
SmolLM2-1.7B-Instruct	24	**6.32**	-0.31	-0.56	0.00	{2, 3}
SmolLM3-3B-Base	36	**5.32**	2.11	-1.03	-2.08	–
SmolLM3-3B	36	**4.26**	1.68	-0.85	-1.65	–
Qwen3.5-0.8B-Base	24	**3.19**	-0.68	-1.03	-0.95	–
Qwen3.5-0.8B	24	**3.07**	-0.67	-1.15	-1.06	–
Qwen3.5-27B	64	**9.61**	**3.18**	1.32	1.49	{4, 51, 52, 60}
Gemma4-E2B	35	0.22	-0.42	-1.27	-1.73	–
Gemma4-31B	60	**5.21**	**6.28**	**3.47**	1.87	{2, 53, 54}

Table 7: Final-layer jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models evaluated on WinoGrande, computed using the cosine-distance metric.

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Llama3.1-8B	32	**4.33**	-0.07	-0.82	-0.92	–
Llama3.1-8B-Instruct	32	**4.26**	0.00	-0.71	-0.81	–
Llama3.1-70B	80	**8.58**	**3.50**	-0.20	-0.29	{2, 4}
Llama3.1-70B-Instruct	80	**11.18**	**3.91**	-0.10	-0.19	{2, 4}
Llama3.2-1B	16	**4.01**	-1.33	-1.70	-1.72	–
Llama3.2-1B-Instruct	16	**3.32**	-1.27	-1.66	-1.67	–
SmolLM2-135M	30	**17.64**	1.55	0.66	-0.76	{2, 25}
SmolLM2-135M-Instruct	30	**11.95**	1.14	0.77	-0.67	{2, 12, 25}
SmolLM2-1.7B	24	**32.06**	-1.43	-1.64	-1.26	{2, 8}
SmolLM2-1.7B-Instruct	24	**17.05**	-1.91	-2.05	-1.54	{2, 8}
SmolLM3-3B-Base	36	2.59	2.24	-0.77	-1.31	–
SmolLM3-3B	36	1.32	1.20	-0.76	-1.17	–
Qwen3.5-0.8B-Base	24	**8.65**	-0.55	-1.19	-0.76	3
Qwen3.5-0.8B	24	**8.56**	-0.45	-1.08	-0.67	3
Qwen3.5-27B	64	**28.17**	**3.13**	0.67	1.02	{51, 52, 55}
Gemma4-E2B	35	0.86	-0.24	-0.95	-1.32	–
Gemma4-31B	60	**27.45**	**16.78**	**13.53**	**3.88**	{2, 53, 54, 56}

Tables 5, 6 show the jump score values on GSM8K using cosine-based displacement and relative L2 displacement, respectively. Tables 7, 8 show the corresponding results on WinoGrande. Both datasets show the same trend as observed on LAMBADA in the main text, with the displacement at the final layer tending to be larger than those at the other layers.

B Influence of massive activation

Massive activation is a phenomenon widely observed in pretrained language models, where specific hidden-state dimensions of a small number of tokens exhibit abnormally large magnitudes. In this section, we separate tokens based on the presence or absence of massive activations and measure their displacement independently. We show that massive activations cannot explain the final-layer jump studied in this work.

Table 8: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$, for open-weight models evaluated on WinoGrande, computed using relative L2 distance.

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Llama3.1-8B	32	**3.52**	0.00	-0.88	-1.04	2
Llama3.1-8B-Instruct	32	**3.18**	0.00	-0.75	-0.92	2
Llama3.1-70B	80	**4.81**	2.29	-0.11	-0.16	4
Llama3.1-70B-Instruct	80	**5.11**	2.16	-0.02	-0.06	{2, 4}
Llama3.2-1B	16	**3.77**	-0.94	-1.41	-1.32	2
Llama3.2-1B-Instruct	16	**3.17**	-0.88	-1.37	-1.31	2
SmolLM2-135M	30	**4.83**	0.73	0.67	-0.73	{2, 3, 12, 25}
SmolLM2-135M-Instruct	30	**4.87**	0.85	0.89	-0.74	{2, 3, 12, 25}
SmolLM2-1.7B	24	**11.44**	-0.83	-0.67	-0.18	{2, 3, 8}
SmolLM2-1.7B-Instruct	24	**6.82**	-0.84	-0.93	-0.33	{2, 8}
SmolLM3-3B-Base	36	**6.81**	2.56	-0.68	-2.00	2
SmolLM3-3B	36	**4.05**	2.50	-0.63	-1.71	2
Qwen3.5-0.8B-Base	24	**5.46**	-0.75	-1.33	-0.60	15
Qwen3.5-0.8B	24	**4.50**	-0.75	-1.37	-0.65	15
Qwen3.5-27B	64	**13.89**	**3.04**	0.77	1.39	{4, 51, 55}
Gemma4-E2B	35	0.17	-0.30	-1.14	-1.84	–
Gemma4-31B	60	**5.42**	**6.02**	**4.64**	2.06	{2, 53, 54}

Table 9: Final-layer jump scores for tokens with and without massive activations.

Model	Tokens with massive activations	ζ_L on Ω_1	ζ_L on Ω_2
Llama-3.1-8B	1.3%	$249,374^\dagger$	**3.16**
Llama-3.1-8B-Instruct	1.3%	$448,387^\dagger$	**3.01**
Llama-3.1-70B	2.1%	$289,595^\dagger$	**5.95**
Llama-3.1-70B-Instruct	2.4%	$119,874^\dagger$	**7.20**
SmolLM2-135M	1.7%	$93.00^\dagger$	**21.13**
SmolLM2-135M-Instruct	2.0%	$32.11^\dagger$	**17.08**
SmolLM2-1.7B	3.8%	$26.68^\dagger$	**40.30**
SmolLM2-1.7B-Instruct	3.8%	$15.78^\dagger$	**14.75**

B.1 Experimental setup

We divide the token set Ω from Definition 1 into tokens with massive activations, Ω_1, and tokens without massive activations, Ω_2, where $\Omega = \Omega_1 \cup \Omega_2$. We measure displacement Ψ_ℓ on Ω_1 and Ω_2, respectively, then compute ζ_L, as defined in Equation (1), for the models in Figure 1. Following Sun et al. (2024), we define a massive activation as one whose absolute value exceeds 100 and is about 1,000 times the median magnitude in the same hidden-state. For each model, we identify the layer–feature-dimension pairs satisfying this criterion.

B.2 Results

Table 9 shows the percentage of tokens exhibiting massive activations and the values of ζ_L computed separately on Ω_1 and Ω_2 for each model [16]. The other experimental settings are the same as those in Section 2.2, and we use LAMBADA as the inference dataset.

For the Qwen3.5 series, no massive activation satisfying this criterion was detected, yet a jump still occurs (Figure 1). For the other models, massive activations were detected, but the final-layer jump

[16] On Ω_1, massive activations cause the hidden-state norm in more than half of the layers to be strongly dominated by a single dimension. Consequently, the vector direction changes very little across layers, making the displacement Ψ_ℓ extremely close to zero. This causes M in Equation (1) to become nearly zero and makes ζ_L numerically unstable. We therefore treat this column only as a reference and base our main comparison on the results for Ω_2.

remained when the evaluation was restricted to Ω_2, the tokens without massive activations. For all 8 models in the table, ζ_L computed on Ω_2 exceeds the threshold of 3.0.

C MODEL SETTINGS

C.1 MODEL PARAMETERS

Table 10: Model settings.

Configuration	1B
Layers	16
Model Dim	2048
FFN Dim	8192
Attention Heads	32
Key / Value Heads	8
Activation	SwiGLU
Weight Tying	True
Rope theta	500000.0
Vocabulary Size	128256

Table 10 summarizes the model settings for the 1B model.

C.2 PARAMETER INITIALIZATION METHOD

The embedding matrix and LM head are initialized from $\mathcal{N}(0, 1/\sqrt{D})$. Each parameter matrix in `Attention` is also initialized from $\mathcal{N}(0, 1/\sqrt{D})$. For the `FFN`, $\boldsymbol{W}_\ell^{\texttt{Up}}$ and $\boldsymbol{W}_\ell^{\texttt{Gt}}$ are initialized from $\mathcal{N}(0, 1/\sqrt{D})$, while $\boldsymbol{W}_\ell^{\texttt{Down}}$ is initialized from $\mathcal{N}(0, 1/\sqrt{D_{\texttt{inter}}})$. The parameters of all `RMSNorm` layers are initialized to 1. After initialization, all parameters are clipped to the range $[-3\sigma, 3\sigma]$, where σ denotes the standard deviation of the corresponding initialization distribution. $\boldsymbol{W}^{\texttt{B}}$ inserted in Section 4 is initialized from $\mathcal{N}(0, 1/\sqrt{D})$, in the same manner as the embedding matrix and LM head. These initialization schemes follow the default settings of the training library `Meta Lingua` (Videau et al., 2024). We use `lm-evaluation-harness` (Gao et al., 2024) for downstream task evaluations.

D TRAINING SETTINGS

D.1 PRE-TRAINING

Table 11: Pre-training settings.

Configuration	1B
lr	3×10^{-4}
global batch	512
sequence len	2048
training step	200,000
training token	200B
weight decay	0.01
epsilon	1×10^{-8}
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)
gradient clip	1.0
scheduler	WSO
warmup	2000

Table 11 summarizes the pre-training settings for the 1B model. For our implementation, we built on the `Meta Lingua` (Videau et al., 2024), using it as the foundation for our pre-training experiments. All pre-training runs were conducted with a random seed of 777. Training a single model required a

Table 12: Jump score under different training settings.

Training Setting		Scheduler	Optimizer	loss	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Baseline	(Section 3.1)	WSO	AdamW	CE	**9.26**	0.39	0.38	0.57	–
Scheduler	(Appendix E.1)	**Cosine**	AdamW	CE	**18.43**	0.67	0.69	1.19	–
		WSD	AdamW	CE	**11.56**	0.42	0.68	0.96	–
Optimizer	(Appendix E.2)	WSO	**Muon**	CE	**7.92**	1.55	0.00	0.24	–
Loss	(Appendix E.3)	WSO	AdamW	**CE+Z-Loss**	**14.19**	2.17	0.56	0.67	–

total computational budget of 28 hours on 32× NVIDIA H100 (80GB) GPUs. The total budget for the experiments reported in the paper is approximately $13,000$ GPU hours.

D.2 Supervised fine-tuning

We split the training data into training and validation sets with a 9:1 ratio. The maximum learning rate is 2×10^{-5}, and we use a linear learning-rate schedule with the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$). All SFT runs were conducted with a random seed of 777. The global batch size is set to 512, resulting in a total of approximately 860 training steps. Training a single model takes about 1 hour using $8\times$ NVIDIA H100 (80GB) GPUs. The total GPU cost for the two SFT runs is about 16 GPU-hours. We use Hugging Face Transformers (Wolf et al., 2020) for training and FastChat (Zheng et al., 2023) for evaluation.

E Detailed Analysis for Identifying the Elements Required for Jumps

In this section, we conduct comprehensive experiments by modifying the training configuration and architectural components to identify the factors that cause the jump in the final layer. For the training configuration, we vary the scheduler (Appendix E.1), optimizer (Appendix E.2), and loss function (Appendix E.3), and show that the phenomenon is not caused by a specific training setup. For the model architecture, we examine the position of Layer Normalization (Appendix E.5), the order of Attention and FFN (Section 3.2), and detailed modifications to only the final-layer FFN (Section 3.3) [17]. Through these experiments, we empirically show that the final-layer jump is caused by two linear layers with sufficiently large rank. Only the components explicitly mentioned are modified; all other settings remain identical to the baseline (Section 3.1).

E.1 Scheduler

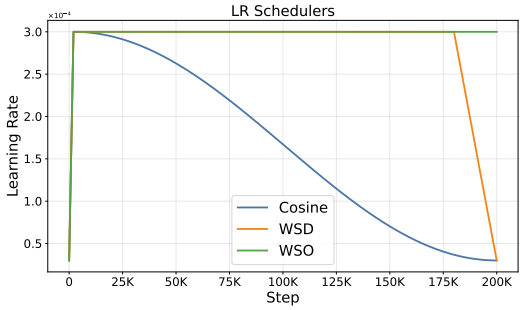


Figure 8: Schedulers

We experimentally demonstrate that the final-layer jump does not depend on the learning rate scheduler by pre-training models with three types of schedulers under the same architectural conditions. The

[17] The order of Attention and FFN and the detailed modifications to the final-layer FFN are discussed in the main text.

three learning rate schedulers used are cosine (Loshchilov & Hutter, 2017), warmup-stable-decay (WSD) (Hu et al., 2024), and Warmup-Stable-Only (WSO) (Yano et al., 2026)[18]. The number of warmup steps is set to 2,000 for all schedulers, and the total number of training steps is 200K, following the baseline setting. The peak learning rate is set to 3×10^{-4}, and for cosine and WSD (i.e., schedulers with decay), the learning rate at the final step is set to 3×10^{-5}. The model configuration and other training settings are identical to those of the baseline.

Figure 2a shows the displacement Ψ_ℓ for models pre-trained with three different schedulers. Table 12 also reports the jump scores. The final-layer jump scores ζ_L for cosine, WSD, and WSO are 18.43, 11.56, and 9.26, respectively. Although the magnitude of the final-layer jump score differs across schedulers, the final-layer jump is consistently observed for all schedulers. This indicates that the final-layer jump does not depend on the scheduler.

E.2 Optimizer

We experimentally demonstrate that the final-layer jump does not depend on the optimizer by pre-training models with two types of optimizers, AdamW (Loshchilov & Hutter, 2019) and Muon (Jordan et al., 2024) under the same architectural conditions. Muon is applied only to two-dimensional parameters excluding the embedding and LM head, while AdamW is applied to all other parameters. The model configuration and other training settings are identical to those of the baseline.

Figure 2a shows the displacement Ψ_ℓ for models pre-trained with two different optimizers. Table 12 also reports the jump scores. The final-layer jump scores ζ_L for AdamW and Muon are 9.26 and 7.92 respectively. Although the magnitude of the jump differs, the jump is consistently observed across all optimizers.

E.3 Z-Loss

We experimentally demonstrate that the jump in the final layer is not caused by excessive scaling of the output logits by pre-training a model with Z-Loss added to the loss function. As shown in Equation (3), since the logits are computed from the final-layer output $\boldsymbol{h}_L$, training near the final layer may proceed in a direction that uniformly increases the values of all logits. Z-Loss is a regularization term introduced to suppress excessive scaling of output logits and to improve numerical stability (OLMo et al., 2025; Chowdhery et al., 2023; Wortsman et al., 2024). Let the logit vector be $\mathbf{z} \in \mathbb{R}^{|\mathcal{V}|}$, where $|\mathcal{V}|$ is the vocabulary size. The Z-Loss is defined as:

$$\mathcal{L}_z = \lambda \left(\log \sum_{i=1}^{|\mathcal{V}|} \exp(z_i) \right)^2 \tag{10}$$

where λ is a hyperparameter that controls the strength of the regularization. This term prevents the overall scale of the logits from growing by penalizing large values of the log-sum-exp. Following OLMo et al. (2025), we set $\lambda = 10^{-4}$.

In our experiments, we used the same Transformer architecture to pre-train two models: one with Z-Loss and one without. For each model, we computed the displacement of hidden-state vectors at each layer, denoted by Ψ_ℓ, and compared the layer-wise changes. Figure 2a shows the progression of Ψ_ℓ and Table 12 shows the jump scores for both models. The final-layer jump scores ζ_L for the models trained with the `CE` loss and the `CE + Z - loss` were 9.26 and 14.19, respectively. In both cases, a sharp increase in Ψ_ℓ is observed at the final layer compared to the preceding layer. Moreover, the displacement at the final layer Ψ_L is significantly larger than that of the previous layer Ψ_{L-1} in both settings, and the rate of increase exhibits a similar trend regardless of whether Z-Loss is applied. These results rule out the tested Z-loss intervention as sufficient to eliminate the jump and are consistent with the architectural evidence presented in later sections.

E.4 With and without LastRMSNorm

We experimentally show that the final-layer jump is not caused by the RMSNorm placed immediately before the LM head layer by pre-training a model in which this RMSNorm is removed. In the baseline

[18]Three learning rate schedulers used are shown in Figure 8

Table 13: Jump score.

Model Setting		ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Baseline	(Equation (3))	**9.26**	0.39	0.38	0.57	–
TTTH	(Equation (11))	**202.02**	-0.37	-1.38	-1.30	–
TRTTH	(Equation (13))	**89.40**	0.31	**32.95**	0.00	–
TTRTH	(Equation (12))	**62.98**	**29.32**	0.00	0.16	–

model, the output of the final layer, $\boldsymbol{h}_L$, is normalized according to its root mean square, followed by a learnable per-dimension scaling, and is then transformed into logits by the LM head Equation (3). In this experiment, we pre-train a model without this RMSNorm, namely,

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{h}_L \tag{11}$$

and examine the trajectory of the hidden-state vectors. The training setup and all other model configurations are identical to those of the baseline.

We show the displacement Ψ_ℓ at each layer in Figure 2, and the jump scores ζ_L, ζ_{L-1}, and ζ_{L-2} in Table 13. In TTTH, the final-layer jump score ζ_L is 202.02, which is large even compared with the baseline model that has an RMSNorm layer immediately before the LM head. This result indicates that the final-layer jump is not caused by the RMSNorm placed immediately before the LM head.

E.5 Position of the $\texttt{RMSNorm}^{\texttt{out}}$

A Transformer applies a normalization layer to the final hidden-state vector $\mathbf{h}_L$. To show that the final-layer jump is not caused by this normalization layer, we conduct experiments by varying both the presence and the position of the normalization layer. Rather than applying $\texttt{RMSNorm}^{\texttt{out}}$ to the final-layer hidden-state vector $\boldsymbol{h}_L$, we apply $\texttt{RMSNorm}^{\texttt{out}}$ to the hidden-state vector $\boldsymbol{h}_{L-1}$ output by the preceding layer and to the hidden-state vector $\boldsymbol{h}_{L-2}$ from two layers earlier, respectively. We denote these configurations as TTRTH ($\texttt{TFL} - \texttt{TFL} - \texttt{RMSNorm} - \texttt{TFL} - \texttt{Head}$) Equation (12) and TRTTH ($\texttt{TFL} - \texttt{RMSNorm} - \texttt{TFL} - \texttt{TFL} - \texttt{Head}$) Equation (13), respectively.

$$\widetilde{\boldsymbol{h}}_{L-1} = \texttt{RMSNorm}^{\texttt{out}}\left(\texttt{TFL}_{L-1}(\boldsymbol{h}_{L-2})\right), \qquad \boldsymbol{z} = \boldsymbol{H}\texttt{TFL}_L(\widetilde{\boldsymbol{h}}_{L-1}) \tag{12}$$

$$\widetilde{\boldsymbol{h}}_{L-2} = \texttt{RMSNorm}^{\texttt{out}}\left(\texttt{TFL}_{L-2}(\boldsymbol{h}_{L-3})\right), \qquad \boldsymbol{z} = \boldsymbol{H}\texttt{TFL}_L(\texttt{TFL}_{L-1}(\widetilde{\boldsymbol{h}}_{L-2})) \tag{13}$$

The training setup and all model configurations not explicitly mentioned are identical to those of the baseline model.

Together with the baseline model, we show the displacement of the hidden-state vectors for these three models in Figure 2. We also show the jump scores ζ_L, ζ_{L-1}, and ζ_{L-2} in Table 13. As seen $\zeta_{L-1} = 29.32 > 3.0$ in TTRTH, and $\zeta_{L-2} = 32.95 > 3.0$ in TRTTH, the hidden-states undergo substantial changes around the RMSNorm layer. On the other hand, the final-layer jump score is consistently large, with values of 62.98 and 89.40 for TTRTH and TRTTH, respectively, indicating that final-layer jumps occur in both cases. These results indicate that the final-layer jump is independent of the position of the RMSNorm layer before the LM head.

E.6 Analysis of jump component: Attention or FFN?

To further localize the source of the jump, we break down the Transformer layer into Attention and FFN components. We decompose the layer-wise displacement Ψ_ℓ between $\boldsymbol{h}_{\ell-1}$ and $\boldsymbol{h}_\ell$ into the displacement in the attention layer, $\Psi_\ell^{\texttt{Attn}}$, and the displacement in the FFN layer, $\Psi_\ell^{\texttt{FFN}}$. Remember that $\boldsymbol{h}_\ell = \texttt{TFL}_\ell(\boldsymbol{h}_{\ell-1}) = \boldsymbol{h}_\ell^{\texttt{Attn}} + \texttt{FFN}_\ell(\boldsymbol{h}_\ell^{\texttt{Attn}})$ and $\boldsymbol{h}_\ell^{\texttt{Attn}} = \boldsymbol{h}_{\ell-1} + \texttt{Attention}_\ell(\boldsymbol{h}_{\ell-1})$ hold in the baseline (Equation (2)). Thus, the decomposed displacements are defined as follows[19]:

$$\Psi_\ell^{\texttt{Attn}} = \mathbb{E}_\Omega\left[\frac{1}{2}(1 - S_C(\boldsymbol{h}_{\ell-1}, \boldsymbol{h}_\ell^{\texttt{Attn}}))\right], \qquad \Psi_\ell^{\texttt{FFN}} = \mathbb{E}_\Omega\left[\frac{1}{2}(1 - S_C(\boldsymbol{h}_\ell^{\texttt{Attn}}, \boldsymbol{h}_\ell))\right]. \tag{14}$$

[19]Note that Ψ_ℓ is not additive over submodules, nor a strict distance metric. In general, $\Psi_\ell^{\texttt{Attn}} + \Psi_\ell^{\texttt{FFN}}$ does not equal Ψ_ℓ.

Similarly, the jump score is also decomposed from a layer-wise metric into metrics at the Attention and FFN component levels $\zeta_\ell^{\texttt{Attn}}$ and $\zeta_\ell^{\texttt{FFN}}$. Let $C = \{\texttt{Attn}, \texttt{FFN}\}$ and $c \in C$.

$$\zeta_\ell^c = \frac{(\Psi_\ell^c - m)}{1.4836\, M}, \quad m = \texttt{median}_{\substack{j=2, \\ c' \in C}}^{L}(\Psi_j^{c'}), \quad M = \texttt{median}_{\substack{j=2, \\ c' \in C}}^{L}(|\Psi_j^{c'} - m|). \tag{15}$$

We show the component-wise analysis of the baseline model. Furthermore, we also train and analyze a reversed-component model, whose Transformer layer consists of FFN and Attention in the reversed order (i.e., FFN→Attention) and defined as

$$\boldsymbol{h}_\ell = \texttt{TFL}_\ell^{\texttt{rev}}(\boldsymbol{h}_{\ell-1}) = \boldsymbol{h}_\ell^{\texttt{FFN}} + \texttt{Attention}_\ell(\boldsymbol{h}_\ell^{\texttt{FFN}}), \text{ where } \boldsymbol{h}_\ell^{\texttt{FFN}} = \boldsymbol{h}_{\ell-1} + \texttt{FFN}_\ell(\boldsymbol{h}_{\ell-1}). \tag{16}$$

We show the decomposed displacement of the baseline model and the reverse model in Figure 2c, and their jump scores in Table 14. In the baseline model, jump score of the final-layer FFN, ζ_L^{FFN}, is $15.21 > 3.0$. In the reverse model, the jump score of the final-layer Attention, ζ_L^{Attn}, is 0.00, whereas jump score of the final-layer FFN, ζ_L^{FFN}, is $16.46 > 3.0$. These experimental results reveal that the jump is not always caused by the component closest to the LM head, but rather by the FFN layer closest to the LM head.

Table 14: Jump score

TransformerLayer Setting	$\zeta_L^{\texttt{FFN}}$	$\zeta_L^{\texttt{Attn}}$	$\zeta_{L-1}^{\texttt{FFN}}$	$\zeta_{L-1}^{\texttt{Attn}}$	$\zeta_{L-2}^{\texttt{FFN}}$	$\zeta_{L-2}^{\texttt{Attn}}$	Other high ζ_ℓ^c
Baseline (Equation (2))	**15.21**	-1.06	-0.21	0.58	0.03	0.00	–
	$\zeta_L^{\texttt{Attn}}$	$\zeta_L^{\texttt{FFN}}$	$\zeta_{L-1}^{\texttt{Attn}}$	$\zeta_{L-1}^{\texttt{FFN}}$	$\zeta_{L-2}^{\texttt{Attn}}$	$\zeta_{L-2}^{\texttt{FFN}}$	
Reverse (Equation (16))	0.00	**16.46**	-0.28	2.40	1.04	2.99	$\zeta_{13}^{\texttt{FFN}}$

E.7 Downstream task performance

While Section 3.2 and the preceding subsection focus on displacement in the Transformer architecture ablations, this subsection examines their downstream task performance. The same downstream tasks as those used to evaluate the pretrained models in Section 4 are considered. Table 15 reports the downstream task performance of the models used in the ablation study. No substantial performance degradation is observed across the ablations.

Table 15: Validation loss and downstream task performance of the models used in our analysis.

Section	Model	Val. Loss ↓	Downstream Performance ↑							
			ARC-E	ARC-C	BoolQ	HellaSwag	OBQA	PIQA	Wino	**Avg.**
Section 3.1	Baseline	2.494	68.6	33.1	59.4	41.8	28.0	70.8	56.0	51.1
Appendix E.1	Cosine	2.445	68.6	33.0	61.4	41.9	25.8	70.3	54.6	50.8
	WSD	2.433	71.1	34.9	60.5	42.5	28.4	72.2	54.4	52.0
Appendix E.2	Muon	2.463	69.0	32.8	58.3	42.4	27.6	71.2	58.6	51.4
Appendix E.3	z-loss	2.477	68.4	32.7	63.1	41.8	29.0	71.9	57.0	52.0
Appendix E.4	TTTH	2.494	69.4	34.0	61.6	41.6	26.0	70.9	58.2	51.7
Appendix E.5	TRTTH	2.500	68.0	31.7	59.1	41.5	26.0	71.7	57.5	50.8
	TTRTH	2.508	68.1	32.3	60.1	41.3	27.0	70.3	56.6	50.8
Section 3.3	WSW	2.498	70.0	34.1	59.6	41.1	25.4	70.6	56.7	51.1
	WRW	2.497	70.0	33.4	61.1	41.0	26.6	72.0	56.7	51.5
	WW	2.503	69.1	33.5	60.2	41.2	27.2	71.2	54.9	51.0
	W	2.503	68.6	32.8	62.4	41.1	28.2	71.3	58.4	51.8

E.8 Hidden-state displacement of our post-trained model

Figure 9 shows the hidden-state displacement of the post-trained model described in Section 4.

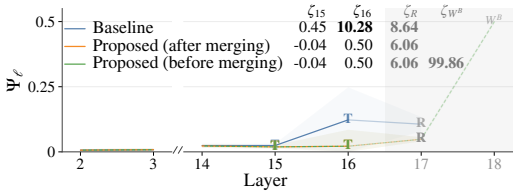


Figure 9: Post-trained models.

Table 16: Downstream performance on LLM-as-a-judge tasks and validation loss after SFT.

Method	Val. Loss ↓	Downstream Performance ↑			
		MT-Bench	Vicuna Bench	WizardLM test	**Average**
Baseline	1.800	1.97	3.99	2.78	2.91
Proposed	1.785	1.96	4.41	2.40	2.92

F Additional Sanity Check of Performance

Table 16 shows downstream performance on three LLM-as-a-judge tasks after SFT. The average performance remains comparable, although individual tasks show increases and decreases.

G Auxiliary Jump Scores for Outside Transformer Layers

The jump score ζ is originally defined based on the displacement Ψ between Transformer layers (Definition 1). In Section 4, to compare transitions outside the Transformer layers on the same scale, we additionally compute the auxiliary scores $\zeta_{L\to R}$ and $\zeta_{R\to \boldsymbol{W}^{\mathrm{B}}}$.

We define Ψ_R as the displacement induced by $\texttt{RMSNorm}^{\texttt{out}}$ and $\Psi_{\boldsymbol{W}^{\mathrm{B}}}$ as the displacement induced by the buffer layer $\boldsymbol{W}^{\mathrm{B}}$. We then compute the corresponding auxiliary jump scores as

$$\zeta_R = \frac{(\Psi_R - m)}{1.4826\, M}, \quad \zeta_{\boldsymbol{W}^{\mathrm{B}}} = \frac{(\Psi_{\boldsymbol{W}^{\mathrm{B}}} - m)}{1.4826\, M}, \tag{17}$$

Here, m and M are the same quantities defined in Equation (1). Ψ_R and $\Psi_{\boldsymbol{W}^{\mathrm{B}}}$ are not included in their computation. Thus, these quantities do not extend the original definition of jump score, but are used only as auxiliary scores for comparing transitions outside the Transformer layers on the same scale.

H Additional Logit Lens Results

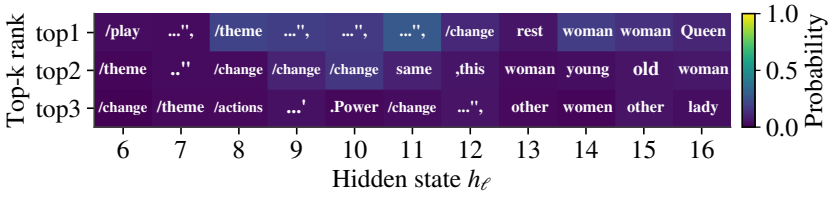
Figures 10–14 show additional Logit Lens results. The proposed model produces tokens that are semantically closer to the final-layer prediction in earlier layers than the baseline model.

I Additional Related Work

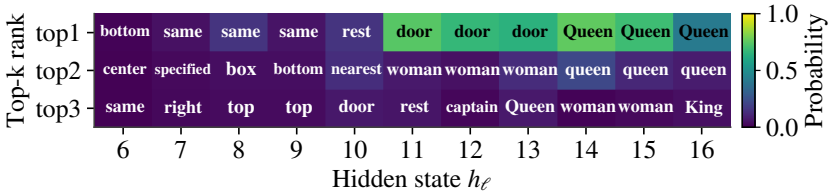
I.1 JREG method

JREG adds a displacement penalty to the cross-entropy objective during pretraining for several layers near the output. As shown in Appendix D in Shibata et al. (2026), constraining only the final layer can simply shift the jump to the preceding, penultimate layer. JREG is therefore designed to constrain multiple layers near the output, which may also alter representations before the final layer compared with standard cross-entropy training.

We directly examined this difference. We pretrained JREG using the same architecture as our baseline model (Table 10) and the same training setup (Table 11), with the hyperparameters reported as the

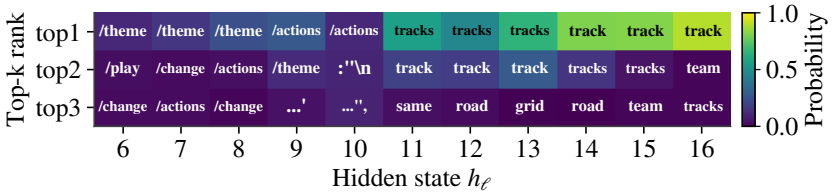


(a) Baseline model

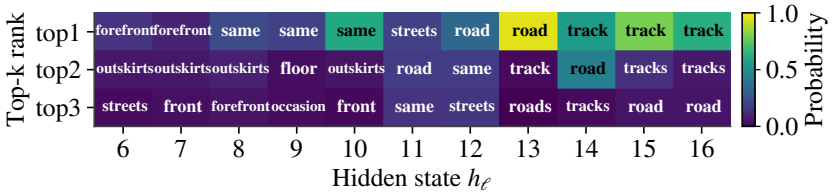


(b) Proposed model (jump-relocated model)

Figure 10: Given the input context ending with "Tom groaned and went inside the carriage to sit down next to the". The gold token is "Queen".



(a) Baseline model



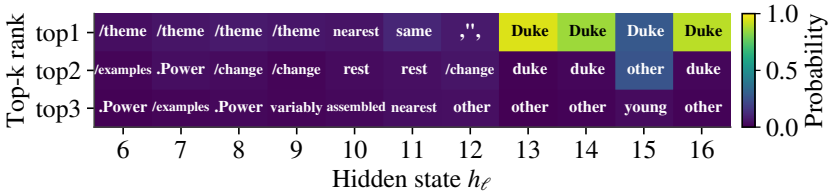
(b) Proposed model (jump-relocated model)

Figure 11: Context ending with "No matter how many times his papa and I tried to discourage him, he found a way back on the", with gold token "track".

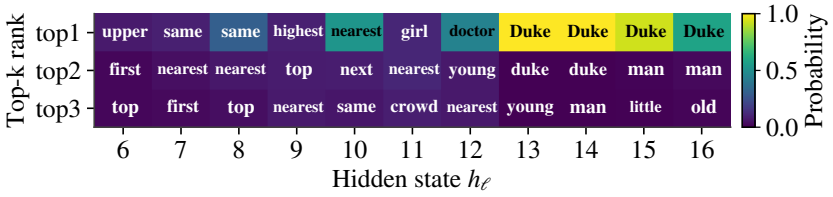
best-performing setting in Shibata et al. (2026), namely $\lambda = 1.0$ and $\alpha = 1.0$. We then measured the layer-wise similarity between the trained model and the baseline using the CKA metric defined in Appendix K.

For JREG, CKA relative to the baseline decreases to 0.71–0.81 at Layers 13–15, while remaining high at 0.93 in the final layer. This result is consistent with JREG suppressing the jump while also inducing changes in the late-layer representations leading to the final layer. In contrast, the jump-relocated model maintains a CKA of approximately 0.95 with the baseline before the final layer, while exhibiting a large difference at the final layer. Thus, compared with JREG, the jump-relocated model reduces late-layer displacement while more strongly preserving the representation geometry before the final layer.

With a JREG-trained model, it is difficult to disentangle whether an observed readout difference and/or the improvement in performance is caused by the reduced jump or by changes in late-layer representations induced by regularization. In contrast, because the buffer-layer model maintains high CKA with the baseline before the final layer, it provides a more controlled comparison for

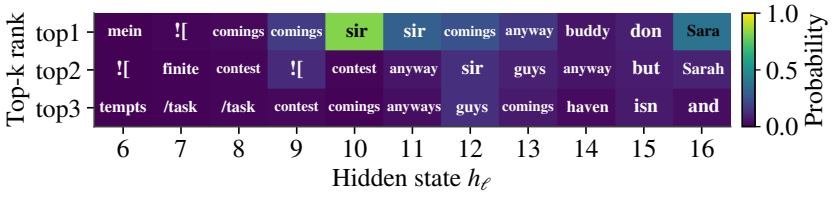


(a) Baseline model

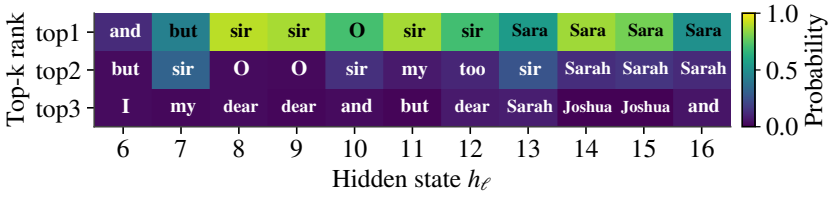


(b) Proposed model ((jump-relocated model)

Figure 12: Context ending with “Sorry,” I said to the”, with gold token “Duke”.



(a) Baseline model



(b) Proposed model ((jump-relocated model)

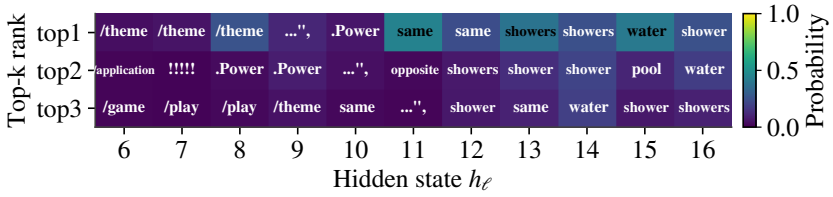
Figure 13: Context ending with “I really, really like you,”, with gold token “Sara”.

our research question (without degrading performance). In this sense, JREG is appropriate for the performance-oriented setting of Shibata et al. (2026), whereas the buffer layer is more appropriate for analyzing the relationship between jump reduction and changes in intermediate readout.

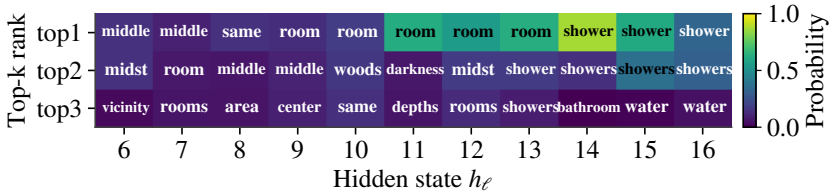
I.2 TunedLens

Tuned Lens is a post-hoc method that learns additional mappings from intermediate hidden-states to final-layer logits of a pretrained model, and it is useful in practice for improving Logit Lens readouts of an existing model. However, it does not directly answer the question studied here: how the jump itself changes the intermediate-layer readouts produced by the model’s native LM head.

One might instead investigate this question using the learned transformations of Tuned Lens. However, because Tuned Lens absorbs the readout mismatch through layer-specific affine transformations, high readout accuracy does not guarantee that the original hidden-state trajectory is smooth. In particular, applying Tuned Lens may obscure the very effect we aim to observe. This parallels the observation of Hewitt & Liang (2019) that a high-capacity probe may solve a task through its own capacity rather than by reading information encoded in the representation. In contrast, we apply the native LM head, without additional training, to matched baseline and jump-relocated models. The purpose of the



(a) Baseline model



(b) Proposed model ((jump-relocated model)

Figure 14: Context ending with "A moment later Toby followed the boy in front of him into the,", with gold token "showers".

Table 17: CKA similarity between the baseline and JREG or Buffer Layer models.

Layer	Baseline vs. JREG (CKA)	Baseline vs. Buffer Layer (CKA; Figure 17)
13	0.81	0.95
14	0.71	0.94
15	0.76	0.92
16 (final layer)	0.93	0.49

buffer layer is therefore not to replace Tuned Lens, but to enable a controlled comparison of the model's native intermediate-layer readouts.

To directly examine the relationship between these approaches, we additionally evaluate both models using Tuned Lens. Following (Belrose et al., 2023), we trained a linear translator with bias for both models on 10M held-out FineWeb-Edu tokens.

Figure 15 reports layer-wise Top-20 overlap under both lenses; the final layer is omitted as it equals Logit Lens. The baseline model with Tuned Lens outperforms the jump-relocated model with the standard Logit Lens at all layers except Layer 15. However, when Tuned Lens is applied to both models, the jump-relocated model performs better at all but three layers. Although Tuned Lens substantially improves the readouts of both models, the advantage of the jump-relocated model remains at most layers even after applying Tuned Lens. This suggests that model-level intervention and a post-hoc translator need not be simple substitutes and may instead be complementary.

J Insert Buffer Layer into Open-Weight Models

In Section 4, we showed that a buffer layer can absorb the final-layer jump from the final Transformer layer during pre-training. In this section, we investigate whether inserting a buffer layer into pre-trained open-weight models and updating only a small subset of parameters can similarly move the final-layer jump to the buffer layer while preserving model performance. Furthermore, for multilingual models, we show that the presence or absence of the final-layer jump changes how the model's internal inference language appears when examined through intermediate hidden-states.

J.1 Model and Training Setting

As shown in Section 3, the final-layer jump is primarily caused by the FFN in the final Transformer layer. Therefore, we restrict the trainable parameters to the FFN in the final Transformer layer,

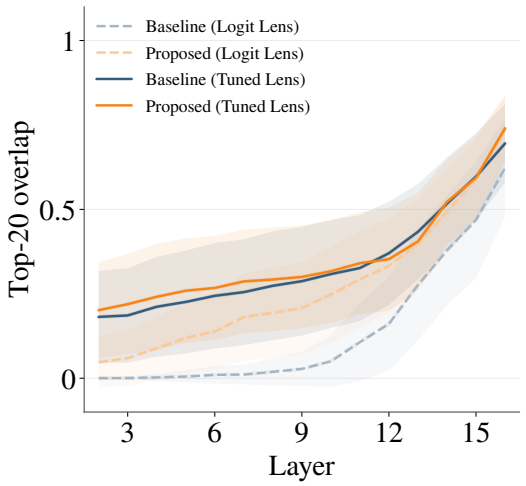


Figure 15: Layer-wise Top-20 overlap for the baseline and proposed models using Logit Lens and Tuned Lens.

the newly inserted buffer layer, and $\texttt{RMSNorm}^{out}$. All other parameters are frozen, including the embedding layer, the LM head, all Transformer layers except the final layer, and the attention module in the final transformer layer.

At the beginning of additional training, we reinitialize the FFN in the final Transformer layer and $\texttt{RMSNorm}^{out}$. Specifically, $\boldsymbol{W}_L^{\texttt{Up}}$ and $\boldsymbol{W}_L^{\texttt{Gt}}$ are initialized from $\mathcal{N}(0, 1/\sqrt{D})$, while $\boldsymbol{W}_L^{\texttt{Dw}}$ is initialized from $\mathcal{N}(0, 1/\sqrt{D_{\texttt{inter}}})$. The parameters of $\texttt{RMSNorm}^{out}$ are initialized to 1. These initialization schemes are identical to those used during pre-training, as described in Appendix C.2.

The buffer layer is inserted between $\texttt{RMSNorm}^{out}$ and the LM head, as in Section 4. However, reinitializing the final-layer FFN and $\texttt{RMSNorm}^{out}$ may substantially change the model output at the beginning of additional training. Therefore, to preserve the output of the pre-trained model as much as possible, we initialize the buffer layer as follows.

Let $\texttt{FFN}'_L$ and $\texttt{RMSNorm}^{out}_{\texttt{init}}$ denote the reinitialized final-layer FFN and output RMSNorm, respectively. We define the input to the LM head after reinitialization as

$$\tilde{\boldsymbol{h}}_L = \boldsymbol{h}_L^{\texttt{Attn}} + \texttt{FFN}'_L\left(\boldsymbol{h}_L^{\texttt{Attn}}\right), \tag{18}$$

$$\tilde{\boldsymbol{h}}_L^{out} = \texttt{RMSNorm}^{out}_{\texttt{init}}\left(\tilde{\boldsymbol{h}}_L\right). \tag{19}$$

In contrast, for the original pre-trained model, the corresponding representation is given by

$$\boldsymbol{h}_L = \boldsymbol{h}_L^{\texttt{Attn}} + \texttt{FFN}_L\left(\boldsymbol{h}_L^{\texttt{Attn}}\right), \tag{20}$$

$$\boldsymbol{h}_L^{out} = \texttt{RMSNorm}^{out}\left(\boldsymbol{h}_L\right). \tag{21}$$

Since the attention module in the final Transformer layer is frozen, $\boldsymbol{h}_L^{\texttt{Attn}}$ is identical before and after reinitialization.

We determine the buffer-layer weight $\boldsymbol{W}^{\texttt{B}}$ by ridge regression using 30K samples equally drawn from Chinese, German, French, and Russian data from FineWeb-2 (Penedo et al., 2025) and English data from FineWeb-Edu, such that the representation transformed by the buffer layer approximates the input to the LM head in the original pre-trained model:

$$\boldsymbol{h}_L^{out} \simeq \boldsymbol{W}^{\texttt{B}}\tilde{\boldsymbol{h}}_L^{out}. \tag{22}$$

This initialization allows the input to the LM head at the beginning of additional training to remain close to that of the original pre-trained model, even after reinitializing the final-layer FFN and $\texttt{RMSNorm}^{out}$.

Table 18: Comparison of downstream task performance between the pre-trained model and the proposed method with buffer-layer insertion and additional training. Although performance varies across tasks, the overall performance is comparable.

Model	ARC-E	ARC-C	BoolQ	HellaSwag	OBQA	PIQA	Winogrande	**Average**
Llama3.2 1B	**66.1**	31.7	**63.7**	**48.1**	27.8	**75.4**	60.3	**53.3**
	64.9	**32.6**	63.5	46.8	**28.2**	74.0	**60.6**	53.0
Llama3.1 8B	**82.3**	52.0	**83.1**	**60.8**	**33.8**	**79.1**	**74.2**	**66.5**
	81.2	**53.0**	82.8	60.0	32.4	**79.1**	73.4	66.0
Qwen3.5 0.8B	**65.7**	33.4	62.5	**39.0**	22.4	**67.9**	**58.7**	**50.0**
	65.3	**34.0**	**63.0**	**39.0**	**23.6**	67.6	57.0	49.9

Table 19: Jump score ζ_L, ζ_{L-1}, ζ_{L-2} and ζ_{L-3}, together with the layers before $L-3$ where $\zeta_\ell > 3$. The final-layer jump disappears in the models with an inserted buffer layer after additional training.

Model	Layer	ζ_L	ζ_{L-1}	ζ_{L-2}	ζ_{L-3}	Other high ζ_ℓ
Llama3.1-8B	32	**3.26**	-0.14	-0.85	-0.95	–
		0.96	-0.14	-0.85	-0.95	–
Llama3.2-1B	16	2.88	-1.25	-1.49	-1.44	–
		-0.40	-1.07	-1.31	-1.26	–
Qwen3.5-0.8B	24	**7.51**	-0.46	-1.04	-0.72	3
		2.40	-0.45	-1.05	-0.74	3

We use Llama3.1-8B, Llama3.2-1B, and Qwen3.5-0.8B as the pre-trained models, all of which are multilingual dense models. For additional training, we use FineWeb-Edu dataset. We train each model for 10K additional steps, corresponding to approximately 2.6B tokens in total. The learning rate is set to 3×10^{-4} for the Llama models and 5×10^{-4} for Qwen3.5-0.8B.

J.2 Performance and hidden-state trajectory

Table 18 shows the downstream task performance before and after additional training. While the performance changes vary across tasks, the overall model performance is maintained without substantial degradation. Table 19 shows the jump score. Models that exhibit a final-layer jump before additional training no longer exhibit the jump after inserting a buffer layer and performing additional training. [20]

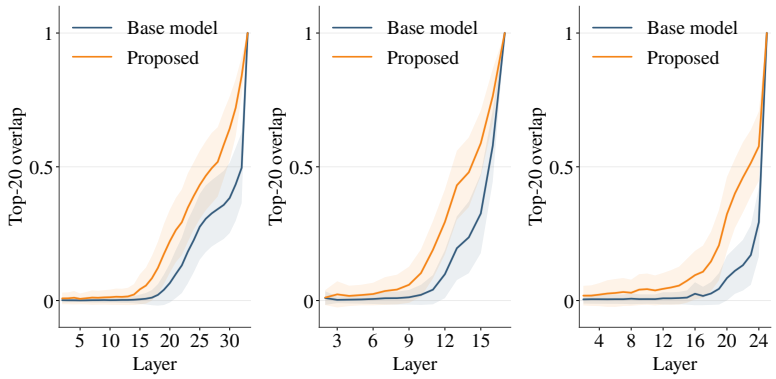
J.3 Top-k overlap

Figure 16 shows the Top-20 overlap for the base models and the models with a buffer layer inserted followed by additional training. As with the models trained with a buffer layer from pre-training, the models with an inserted buffer layer show higher Top-20 overlap than the base models across all intermediate layers. Since the parameters before the final layer are identical between the base and proposed models, their hidden-states are also identical at all intermediate layers. Despite having identical intermediate hidden-states, absorbing the final-layer jump into the buffer layer consistently increases the Top-20 overlap with the final layer across all intermediate layers. This result suggests that the presence or absence of the final-layer jump can influence the layer-wise predictions observed through the Logit Lens.

J.4 MULTILINGUAL ANALYSIS

Previous work using Logit Lens has reported that multilingual language models pass through English-like abstract representations in intermediate layers and project them into the target language near the final layers (Wendler et al., 2024). However, because this analysis relies on Logit Lens, the

[20] Although Llama3.2-1B already had $\zeta_L < 3.0$ before buffer insertion, its jump score further decreased from 2.88 to -0.40 after inserting the buffer layer and performing additional training.



(a) Llama3.1-8B (b) Llama3.2-1B (c) Qwen3.5-0.8B

Figure 16: Top-20 overlap across layers. Base model denotes the original open-weight model, while Proposed denotes the model with an inserted buffer layer followed by additional training. The Proposed model consistently shows higher Top-20 overlap across all intermediate layers.

observed layer-wise behavior may be affected by the final-layer jump. Therefore, in this section, we conduct the same analysis on open-weight models in which a Buffer layer is inserted and the final Transformer-layer jump is relocated to the buffer layer, and investigate how the layer-wise behavior observed with Logit Lens changes.

We follow the experimental setup of (Wendler et al., 2024), using a task that translates non-English words into Chinese. An example prompt used for inference is shown below.

Français: “nuage” - 中文: “云”
Français: “sac” - 中文: “包”
Français: “bouche” - 中文: “口”
Français: “montagne” - 中文: “山”
Français: “cœur” - 中文: “...”

We decode the hidden-state $\boldsymbol{h}_\ell$ at each layer using Logit Lens and measure the probabilities assigned to the corresponding correct Chinese and English tokens. We then average these probabilities over all prompts. We use these probabilities as an indicator of whether the internal representation is closer to Chinese or English. The inference dataset is the same as that used in (Wendler et al., 2024).

K Centered Kernel Alignmen (CKA)

One might think that the noisy Logit Lens outputs of the baseline model reflect poor intermediate representations, rather than a readout mismatch with the final LM head. To examine this possibility at the level of representation geometry, we compare the baseline and proposed models using linear Centered Kernel Alignment (CKA) (Kornblith et al., 2019). For sample-centered hidden-state representations $X \in \mathbb{R}^{n\times D}$ and $Y \in \mathbb{R}^{n\times D}$ computed from $n = 5153$ tokens, we compute:

$$\mathrm{CKA}(X,Y) = \frac{\|X^\top Y\|_F^2}{\|X^\top X\|_F \|Y^\top Y\|_F}. \qquad (23)$$

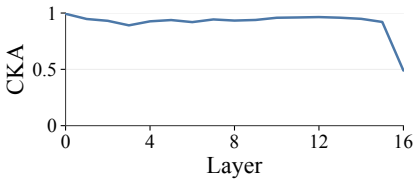


Figure 17: CKA similarities between the baseline and proposed model.

This is equivalent to comparing the sample-wise similarity matrices $XX^\top$ and $YY^\top$, so it captures similarity in representation geometry rather than direct alignment of coordinate axes. As shown in Figure 17, the CKA values remain close to one across most intermediate layers. Thus, the baseline model remains close to the proposed model in intermediate-layer representation geometry, despite the large difference in their Logit Lens readouts. This suggests that the noisy Logit Lens outputs are not simply due to poor intermediate representations, but are better understood as a mismatch between the intermediate states and the LM head.

L LICENSES FOR EXISTING ASSETS

We list the licenses and terms of use for the existing assets used in this work.

Training libraries. We use `Meta Lingua` (Videau et al., 2024) as the codebase for our pre-training experiments. `Meta Lingua` is released under the BSD-3-Clause license. We also use Hugging Face Transformers (Wolf et al., 2020), which is released under the Apache-2.0 license.

Evaluation libraries. For downstream task evaluations, we use `lm-evaluation-harness` (Gao et al., 2024), which is released under the MIT license. We also use FastChat (Zheng et al., 2023), which is released under the Apache-2.0 license.

Training datasets. We use FineWeb-Edu (Penedo et al., 2024) and the Tulu-v1 SFT mixture (Wang et al., 2023) as training data. These datasets are distributed under licenses from the Open Data Commons Attribution license family.

Open-weight models. We use several existing open-weight models for comparison and analysis. Llama 3.1 and Llama 3.2 (Grattafiori et al., 2024) are governed by the Llama 3.1 Community License Agreement and the Llama 3.2 Community License Agreement, respectively. SmolLM2 (Allal et al., 2025), SmolLM3 (Bakouch et al., 2025), Qwen3.5 (Qwen Team, 2026), and Gemma4 are released under the Apache-2.0 license.

We use all assets in accordance with their respective licenses and terms of use, and cite the original sources where appropriate.