

The Feasibility of Token-Level Compute Allocation across Depth in Pretrained Transformers

Stanford CS224N Custom Project

Anjali Sreenivas
Department of Computer Science
Stanford University
anjalisr@stanford.edu

Yuchen Li
Department of Physics
Department of Computer Science
Stanford University
yucli@stanford.edu

Abstract

The inference cost of Transformer-based LLMs has motivated conditional computation methods that activate only a subset of parameters for each input, but these architectures are typically trained from scratch. Here, we study whether a pre-trained dense Transformer exhibits token-level variation in depth usage that could support post-hoc dynamic compute allocation. Using TunedLens on Pythia-160M evaluated on WikiText-103, we define a notion of token stabilization based on when intermediate-layer next-token predictions persistently match the final-layer prediction. Many tokens stabilize well before the final layer, and we investigate whether these early-stabilizing ("easy") tokens also have simpler hidden-state trajectories that might justify skipping deeper computation. We find that simple geometric measures, such as inter-layer cosine similarity and hidden-state update norms, do not clearly distinguish easy tokens from those that never stabilize in the vocab space ("hard" tokens). Furthermore, translated hidden-state updates for both classes lie largely in logit-relevant directions. However, easy tokens are somewhat more predictable in hidden space: their layer-to-layer updates are better approximated by affine maps, and linear probes can classify easy versus hard tokens from hidden states with about 80% accuracy. These results suggest that pretrained Transformers contain subtle differentiating signals related to token-wise depth usage, indicating potential for early exit, though reliably exploiting these signals may require more sophisticated methods.

1 Key Information to include

- Mentor: Julie Kallini
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any): 3

Team member	Late days remaining
Yuchen Li	3
Anjali Sreenivas	3
Total pooled	6
Team size	2
Late days applied for the report	3

2 Introduction

Large language models using traditional dense Transformer architectures incur high inference costs because the same sequence of layers is applied to every token, even though some tokens are much easier to predict than others. This motivates conditional computation over model depth: if some tokens can be resolved with fewer compute-intensive layers, then inference could become more efficient without reducing model capability. Recent work such as Mixture-of-Depths [1] and Dynamic Token Routing [2] show that Transformer language models can be trained to allocate computation selectively across tokens and layers under a fixed compute budget. These methods typically build conditional behavior into the architecture during training.

In this work, we study a complementary question: to what extent do *pretrained* Transformers already exhibit token-level variation in depth usage? If so, pretrained models may contain signals in their hidden states that could support depth-adaptive computation without retraining from scratch. More broadly, such signals may help clarify how depth is used during inference and whether it is utilized uniformly across tokens.

To study this question, we adopt the view of Transformers as performing iterative refinement of a next-token prediction over layers. Using TunedLens [3], we decode intermediate hidden states into the model’s final prediction space and track how token predictions evolve across depth. This gives a concrete notion of token stabilization, as some tokens appear to reach their final prediction much earlier than others. That observation is suggestive for conditional computation, since it raises the possibility that different tokens may have different effective depth requirements even in a dense pretrained model. Because routing methods ultimately decide how to modify token hidden states, we then test a series of hypotheses about how hidden states evolve for tokens with different levels of stabilization.

3 Related Work

Conditional computation over depth. The most directly related conditional compute work is Mixture-of-Depths (MoD) [1], which learns to route only a subset of tokens through each layer under a fixed compute budget. MoD shows that token-wise depth allocation can be effective, but it does so in a model trained specifically for sparse depth usage. Here, we do not train a new architecture, and instead ask whether a standard pretrained Transformer already exhibits signals consistent with token-wise variation in depth usage.

Early exiting and layer skipping. A closely related line of work studies early exiting and layer skipping at inference time. CALM [4] uses next token confidence criteria to adapt computation during generation, while LayerSkip [5] introduces training modifications that make earlier layers more predictive, enabling early exiting. Recent work on adaptive layer skipping in pretrained LLMs [6, 7] pushes even closer to our setting by asking how skipping can be applied without training a sparse model from scratch. Our project is more diagnostic than algorithmic: rather than proposing a skipping rule, we ask what properties of pretrained inference trajectories would justify such rules in the first place.

How LLMs use their depth. Our framing also relates to recent papers that study how depth is used in pretrained language models. [8] argues that later layers often contribute less than earlier ones, that skipping later layers has smaller effects on future computations and predictions, and that deeper models may spread similar computations over more layers rather than learning fundamentally new ones. [9] presents a complementary “guess-then-refine” view, where early layers favor high-frequency statistical guesses and later layers refine them using context; it also finds that depth usage varies by task and token type.

4 Approach

We use Pythia-160M (deduped) [10], a pretrained decoder-only Transformer, for our experiments. The model is evaluated over a subset of the test split of WikiText-103-raw, a dataset of English Wikipedia articles. The data is preprocessed by filtering out texts with excessive punctuation, numeric tokens, or capitalization; this yields a total of 1605 sequences and 2.6×10^5 tokens.

We evaluate how pretrained Transformers use their depth by considering each layer as an iterative refinement to the predicted logits over the vocabulary. However, Transformers are not trained *a priori* to produce logits from hidden state representations at intermediate layers. One simple way to bypass this challenge is LogitLens [11], which computes for each layer ℓ the logits

$$\text{LogitLens}[\mathbf{h}_\ell] = \mathbf{v}_\ell = \text{LayerNorm}[\mathbf{h}_\ell]W_U \quad (1)$$

where $\mathbf{h}_\ell \in \mathbb{R}^d$ is the hidden state at layer ℓ and $W_U \in \mathbb{R}^{d \times |\mathcal{V}|}$ is the unembedding matrix. This can be interpreted as the predicted token if the contributions to the residual stream from all later layers $\ell' > \ell$ were zero.

TunedLens [3] is a refinement of LogitLens that applies a learned affine mapping before the final LayerNorm:

$$\text{TunedLens}[\mathbf{h}_\ell] = \text{LogitLens}[A_\ell \mathbf{h}_\ell + \mathbf{b}_\ell], \quad (2)$$

where $A_\ell \in \mathbb{R}^{d \times d}$, $\mathbf{b}_\ell \in \mathbb{R}^d$ are learned to minimize the KL divergence between the intermediate layer vocab distribution and the final layer vocab distribution. This is motivated by the chance to apply a change of basis between intermediate layer representations and final layer representations, and empirically has been shown to produce more consistently interpretable results. In the following section, we use the pretrained TunedLens (available from HuggingFace) for the Pythia 160M model to analyze token evolutions over model depth.

5 Results

5.1 TunedLens defines a notion of token stabilization in model depth

Here, we use TunedLens to motivate the potential of depth-adaptive routing or early exiting mechanisms in pretrained Transformers. Figure 1a illustrates the result of TunedLens applied on a sample sequence. This example shows some tokens, such as ‘a’ and punctuation, which stabilize in shallow layers. Moreover, the KL divergence $\text{KL}(p_\ell, p_L)$ between the vocab distribution at layer ℓ and the vocab distribution at the final layer L remains low for all $\ell < L$ for these tokens. Also shown are tokens, such as the one following ‘and’, whose representation shifts quite significantly over the depth of the Transformer, and whose KL divergence decreases significantly over the depth. These observations suggest that pretrained Transformers may process different tokens over different effective depths.

In Figure 1b-c, we plot the token-level cross entropy of the intermediate layer logits and the token-level forward KL divergence $\text{KL}(p_\ell, p_L)$ over the WikiText test corpus. We highlight tokens with early stabilization by subsetting the decile with the lowest respective cross entropy or KL divergence at layer 1 and track their trajectory over the full depth (purple). These tokens have intermediate-layer logits that persist with low cross entropy and low KL divergence with the final layer logits, supporting this notion of “easy” versus “hard” tokens across the corpus.

Finally, we consider a simple way of classifying tokens by the depth after which they “stabilize” according to TunedLens. We define a token to be stable under TunedLens at layer ℓ if for all layers $k \geq \ell$, the argmax of p_k is equal to the argmax of p_L (i.e., top-1 token is consistent). Figure 1d shows that the majority of tokens stabilize before the final layer L , with some stabilizing as early as the raw embeddings (layer 0). Indeed, some examples of this phenomenon are illustrated in Figure 1a.

These results indicate that the TunedLens probe defines a notion for token stabilization through model depth which correlates intuitively with the cross entropy of that token. However, conditional compute strategies such as early exiting that skip parts (or the entirety) of Transformer blocks must define a corresponding way of how to update the *hidden states* over Transformer blocks that are skipped. Do tokens that stabilize under TunedLens have a hidden state evolution that are distinct from tokens that do not? Are these hidden state trajectories more predictable for early-stabilizing tokens? We address these questions in the following sections.

5.2 Hidden state cosine similarities do not indicate token stabilization

Perhaps the *simplest* hypothesis for what separates the hidden state trajectories of tokens that stabilize early versus late is that hidden state updates $\Delta \mathbf{h}$ go to zero for tokens that stabilize early. Indeed, this is the design choice adopted by many early-exiting Transformers trained from scratch [4], as the hidden states of early-exited tokens are directly copied to deeper layers.

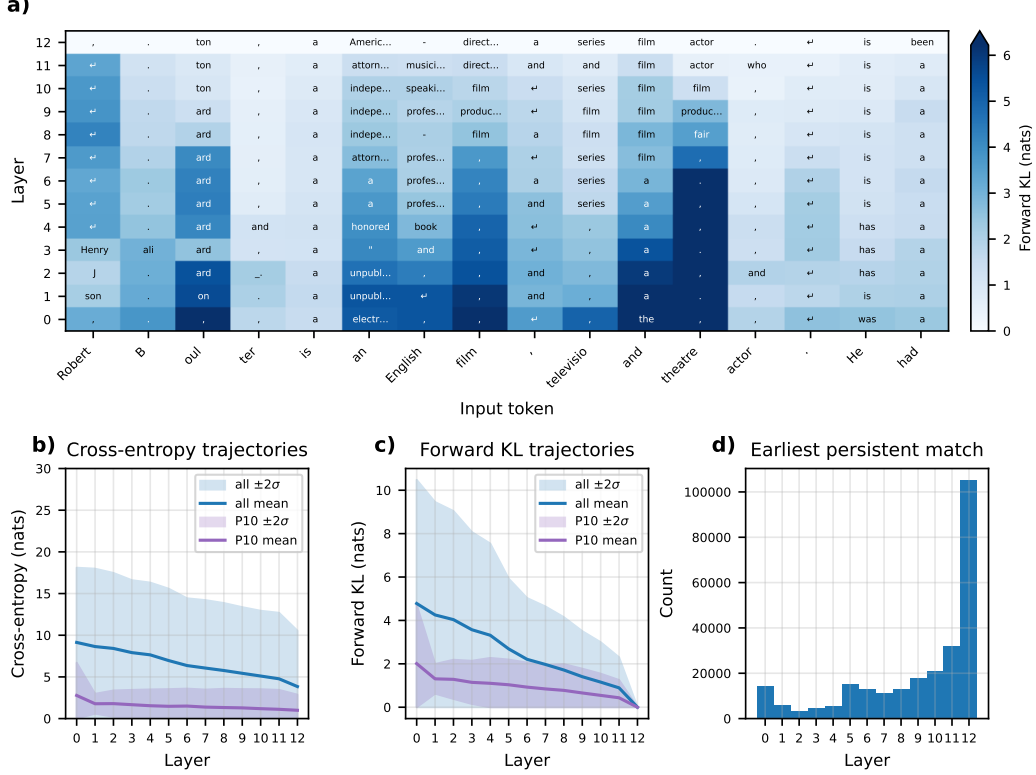


Figure 1: TunedLens prediction trajectories for the Pythia-160M model evaluated on the WikiText-103 test set. (a) Per-layer top predicted token for an example sequence, colored by forward KL divergence to the final-layer distribution. (b, c) Cross-entropy and forward KL divergence as a function of layer, computed per token across all samples. The blue band shows the mean ± 2 standard deviations; the purple band highlights the bottom decile (P10) of tokens ranked by layer-1 cross-entropy/KL, illustrating tokens whose representations stabilize early and remain low throughout. (d) Histogram of the earliest layer at which each token’s top prediction persists unchanged through to the final layer.

We test this hypothesis on our pretrained transformer by computing cosine similarities between hidden states at different depths $\cos(\mathbf{h}_i, \mathbf{h}_j) = \frac{\langle \mathbf{h}_i, \mathbf{h}_j \rangle}{\|\mathbf{h}_i\| \|\mathbf{h}_j\|}$. If hidden state updates go to zero for tokens that stabilize, the cosine similarity between intermediate-layer hidden states with final-layer hidden states for tokens that stabilize should be higher than tokens that do not.

Figure 2a-c shows that this is not the case. Tokens that never stabilize have intermediate hidden states whose cosine similarity with deeper hidden states decays roughly linearly with depth, until a drop to near-zero at the final layer (Figure 2a). Tokens which stabilize early actually have hidden states whose cosine similarities decays more *strongly* with the depth of the model, which is opposite to what the hypothesis above would imply.

5.2.1 TunedLens-translated hidden states

Since TunedLens nevertheless identifies tokens that stabilize early in model depth, we consider the same cosine similarity analysis, but this time using the image of the hidden states under the TunedLens affine mapping. Specifically, for tokens with hidden state $\mathbf{h}_\ell$, we compute $\mathbf{h}'_\ell = \text{LayerNorm}_L[A_\ell \mathbf{h}_\ell + \mathbf{b}_\ell]$ where A_ℓ and $\mathbf{b}_\ell$ are the TunedLens weights for layer ℓ and $\text{LayerNorm}_L[\cdot]$ is the final LayerNorm. We refer to $\mathbf{h}'_\ell$ as the *translated* hidden states, as this affine mapping “translates” the hidden state at layer ℓ so that they can be unembedded with W_U .

Figure 2d-f shows that while the cosine similarities between translated hidden states are typically higher than raw hidden states, there is no significant difference in the cosine similarities for tokens which stabilize early. Another somewhat perplexing fact is that the inter-layer cosine similarities for

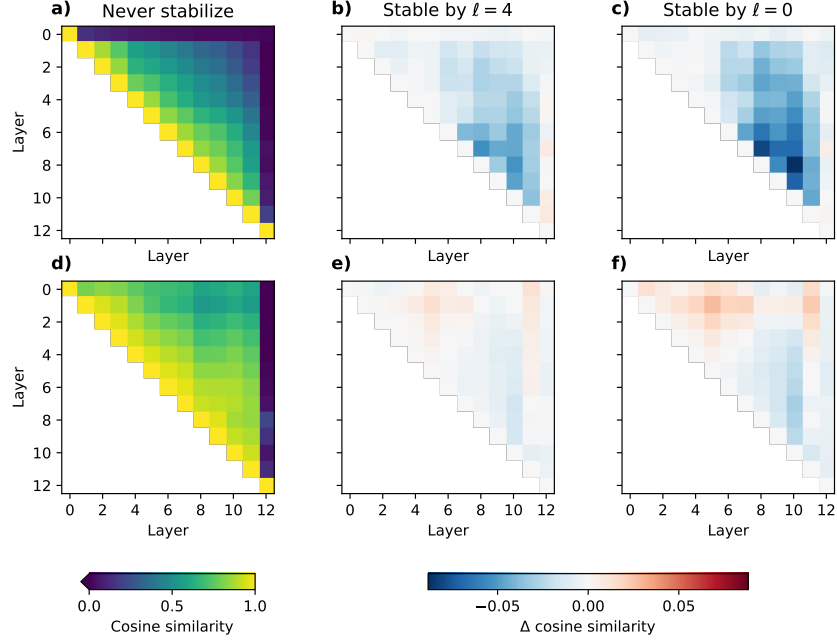


Figure 2: (a) Average cosine similarities $\cos(\mathbf{h}_i, \mathbf{h}_j)$ between hidden states at layer i and j for the tokens that never stabilize according to TunedLens. (b) As in (a), but calculated over tokens that stabilize by layer 4. The plotted colormap shows the difference from (a). (c) As in (a), but calculated over tokens that stabilize by layer 0. (d-f) Same as (a-c), but with cosine similarities $\cos(\mathbf{h}'_i, \mathbf{h}'_j)$ between *TunedLens*-translated hidden states $\mathbf{h}'$.

all stabilization classes remain somewhat high (> 0.5) until the final layer for which it drops to near 0. This is surprising because the TunedLens affine mapping is interpreted to transform intermediate layer hidden states into the final layer’s basis so that they can be unembedded.

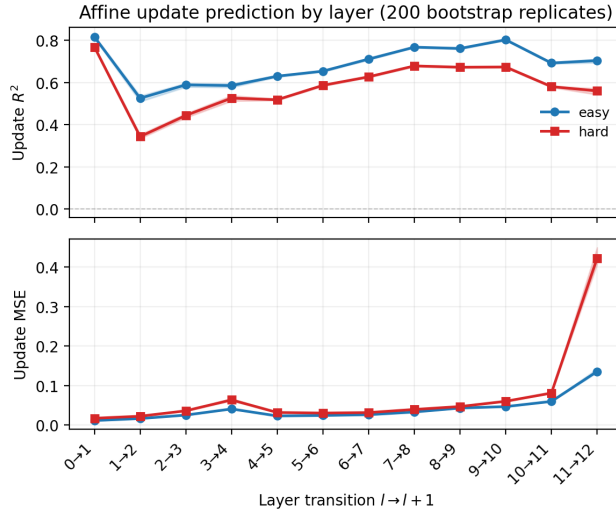


Figure 3: Accuracy of learned affine mappings to predict layer-to-layer hidden state updates on unseen easy and hard tokens. The per-layer median and 95% bootstrap confidence interval bands (narrow) are shown for both R^2 and MSE.

5.3 Hidden state updates for depth-stable tokens are more predictable

5.3.1 Layer-to-layer affine mapping of hidden states

Since cosine similarity between hidden states diverges similarly over model depth across token stabilization classes, we turn to other properties of the hidden space could distinguish the easy tokens. One hypothesis is that easy tokens follow a simpler hidden-state trajectory, which might be well-approximated by learned affine transformations between layers.

For each layer transition $\ell \rightarrow \ell + 1$, we learn an affine transformation $W_\ell \in \mathbb{R}^{d \times d}$, $\mathbf{a}_\ell \in \mathbb{R}^d$ by minimizing

$$\mathcal{L}_\ell = \sum_i \left\| \Delta \mathbf{h}_\ell^{(i)} - (W_\ell \mathbf{h}_\ell^{(i)} + \mathbf{a}_\ell) \right\|_2^2,$$

where $\Delta \mathbf{h}_\ell^{(i)}$ is the residual update for token i from layer $\ell \rightarrow \ell + 1$, and $\mathbf{h}_\ell^{(i)} \in \mathbb{R}^d$ is its hidden state at layer ℓ . Separate weights are fit for the easy and hard token classes, and we subsample both categories to equal size ($N = 14353$ tokens) to avoid class imbalance. We use a simple bootstrap to estimate uncertainty over token resamplings (Appendix 8.2).

Figure 3 shows that across all layer transitions, the easy token class achieves higher R^2 values than the hard token class, indicating that the learned affine mappings produce predictions that are more strongly correlated with the true updates $\Delta \mathbf{h}_\ell^{(i)}$. Easy tokens also exhibit slightly lower mean-squared error across most layer transitions, though the difference is small except for in the final transition from layer 11 $\rightarrow$ 12.

5.3.2 Learning a simple probe to distinguish “easy” and “hard” tokens

Many dynamic token-level compute allocation methods rely on a learned router to determine, at each layer, how each token should be processed [2, 7]. In the context of pretrained dense Transformers, this raises a simpler preliminary question: do token representations already contain information that separates easy from hard tokens? To investigate this, we train logistic regression probes on the hidden states of Pythia-160M and measure how accurately they recover this token classification across model depth.

Specifically, we train 13 logistic regression classifiers—one on the embedding layer and one on each of the L hidden layers—using $N = 14353$ token samples drawn from both the easy and hard token classes, with 5-fold cross-validation. Figure 4 shows that the probes achieve roughly 80% accuracy in predicting the token class from the representation at each layer. Performance is largely consistent across depth.

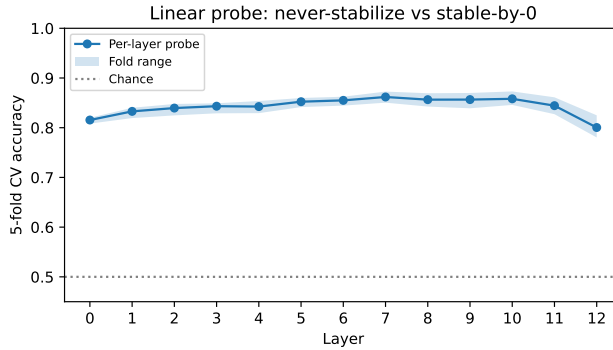


Figure 4: Binary accuracy of linear probes trained to predict, from hidden states at layer ℓ , whether a token is in the “never stabilize” (hard) or “stabilize by layer 0” (easy) class.

5.4 Do layer-to-layer hidden state updates point toward logit-relevant directions?

Finally, because changes between logits Δl can be expressed in terms of the unembedding of changes in TunedLens-translated hidden states, $\Delta l = l_{\ell+1} - l_\ell = W_U(\mathbf{h}'_{\ell+1} - \mathbf{h}'_\ell) = W_U \Delta \mathbf{h}'_\ell$, we explore

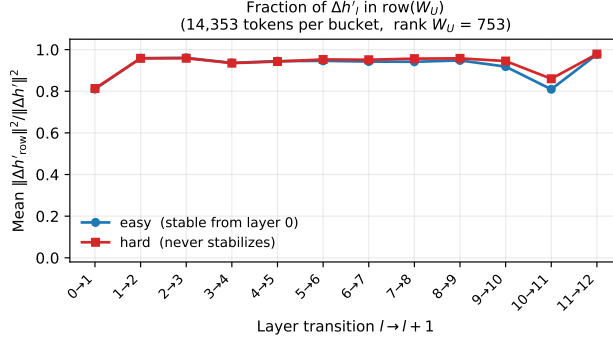


Figure 5: Proportion of hidden state update vector within the row space of the unembedding matrix. Addresses the question, do layer-to-layer hidden state updates point toward logit-relevant directions? Values close to 1 indicate that most of the update lies in directions that can affect the logits, while values close to 0 indicate that the update primarily lies in directions ignored by the unembedding matrix.

whether the magnitude or direction of $\Delta \mathbf{h}'_\ell$ can explain the asymmetry in the vocab space between when easy and hard tokens stabilize. We find that update norms $\|\Delta \mathbf{h}'_\ell\|$ are nearly identical for easy (stabilizing) and hard (non-stabilizing) tokens (Appendix Figure 6), indicating that magnitude alone does not explain the difference.

Next, we examine whether the translated updates $\Delta \mathbf{h}'_\ell$ align with directions that affect logits. Decomposing each update into its row space and null space components with respect to W_U (Appendix section 8.1), we find that most of the vector lies in the row space for both stabilizing and non-stabilizing tokens, shown in Figure 5. A finer decomposition along the singular directions of W_U shows nearly identical spectra (Appendix Figure 8), even considering the dominance of $\sigma_1 \approx 776 \gg \sigma_{k>1}$ (Appendix Figure 7).

Together, these results suggest that the layer-wise hidden state changes for both easy and hard tokens affect the logit space in similar ways. Therefore, the difference in stabilization cannot be easily attributed to the geometric structure of the updates.

6 Discussion

Our results reveal an interesting dissociation: inter-token stabilization differences are visible in the TunedLens vocabulary space, but are surprisingly difficult to map to hidden-state dynamics directly. Qualitatively, the easy tokens that stabilize early in the vocab space tend to be simple words and punctuation—tokens requiring little contextual refinement—while hard tokens are content-rich or ambiguous. But in the hidden space, we find that both token classes undergo hidden state updates of similar magnitudes which point similarly in logit-relevant directions. The difference appears to be not so much in *how* the transformer architecture updates tokens, but more so in *where* those updates land.

This has a direct implication for dynamic compute allocation. Naive early-exit heuristics based purely on hidden-state similarity are unlikely to succeed on pretrained models: hidden-state trajectories do not converge for either easy or hard tokens, making simple freezing strategies ineffective. Although easy and hard tokens are somewhat linearly separable in hidden-state space, their distinction is subtle: hidden update norms, directions, and spectral profiles are near-identical across the two classes, so straightforward geometric criterion may not reliably indicate when a token is sufficiently processed.

More promising signals arose from simple learned probes: linear classifiers could separate easy and hard tokens decently well, and layer-to-layer affine mappings predicted hidden-state updates for easy tokens more accurately. These suggest that routing mechanisms for pretrained transformers could warrant further exploration, but because the signals we gathered in the hidden space are relatively weak, it’s also likely that robust dynamic compute allocation techniques will require fine-tuning.

7 Conclusion

We investigated whether pretrained Transformers exhibit signals that could support token-level dynamic compute allocation without retraining. Our results show that: (1) some tokens stabilize early in the TunedLens vocabulary space, suggesting potential for reduced computation; (2) hidden-state trajectories are difficult to distinguish between easy and hard tokens using simple metrics like cosine similarity, but affine maps and linear probes provide partial predictability and differentiating capability; and (3) the layer-to-layer updates to hidden state vectors in the logit-ready basis lie largely in logit-relevant directions for all tokens.

One limitation is our use of the comparatively small Pythia-160M model on a small corpus. The results here should therefore be taken with a grain of salt when considering depth use in larger language models. Future work should test these hypotheses on larger models and on a more diverse test corpus.

Another limitation of our approach is that we generally focus on conducting simpler analyses before more complex ones. For example, while we found that cosine similarities over the entire hidden space fail to elucidate differences in hidden state trajectories for easy versus difficult tokens, it’s possible that another metric may have higher distinguishing capability. Additionally, differences between the hidden state updates (Section 5.4) may be small but nevertheless significant when mapped into the logit space.

Overall, our results show modest promise for post-hoc depth adaptation, but the relevant signals are subtle. Future work could extend our layer-to-layer affine mappings of hidden state updates to evaluate the effectiveness of layer skipping for easy tokens in pretrained Transformer. However, robust depth allocation strategies will likely require finetuning so that the model explicitly learns to differentiate easy and hard token trajectories at the hidden-state level.

Team contributions (Required for multi-person team)

Anjali did the layer-to-layer affine transformation experiment (5.3.1), and SVD analysis of the unembedding (5.4). Yuchen did the initial TunedLens token stabilization analysis (5.1), and the linear probe (5.3.3). Both worked on the hidden state cosine similarity analysis (5.2) and contributed equally to this writeup.

Acknowledgements

We thank our mentor, Julie Kallini, for her encouragement and feedback, as well as our professors, Prof. Diyi Yang and Prof. Yejin Choi, for their support.

8 Appendix

8.1 Full analysis for understanding layer-to-layer hidden state updates in terms of logit-relevant directions

Logit-ready updates. The raw hidden state $\mathbf{h}_\ell \in \mathbb{R}^d$ at layer ℓ is not aligned with the space used by W_U . We apply TunedLens’ learned affine probe followed by layer normalization to obtain the logit-ready representation of each hidden state

$$\mathbf{h}'_\ell = \text{LayerNorm}_L[A_\ell \mathbf{h}_\ell + \mathbf{b}_\ell], \quad \text{such that logits} \approx W_U \mathbf{h}'_\ell.$$

We then analyze layer-to-layer changes in this final layer space: $\Delta \mathbf{h}'_\ell = \mathbf{h}'_{\ell+1} - \mathbf{h}'_\ell$. For the final layer no probe is needed; we apply the final LayerNorm directly.

Row-space decomposition. The SVD $W_U = U\Sigma V^\top$ provides an orthonormal basis $V = [v_1, \dots, v_H]$ for the hidden-state space, ordered by how strongly each direction influences the logits. Let r be the rank of W_U ; then the first r vectors span the row space of W_U (logit-relevant directions), and the remaining $H - r$ vectors form its null space (W_U maps these to zero). Empirically, we found that $r = 753$ out of $H = 768$ for W_U , leaving 15 null-space dimensions.

To measure how much of the update lies in logit-relevant directions, we express the translated update $\Delta \mathbf{h}'_\ell$ in this basis:

$$\Delta \mathbf{h}'_\ell = \sum_{i=1}^H c_i v_i, \quad c_i = \Delta \mathbf{h}'_\ell \cdot v_i.$$

The component of the update that lies in the row space (the first r directions) is

$$\Delta \mathbf{h}'_{\ell, \text{row}} = \sum_{i=1}^r c_i v_i.$$

We then measure the fraction of the update lying in the row space as

$$\text{frac}_{\text{row}}(\Delta \mathbf{h}'_\ell) = \frac{\|\Delta \mathbf{h}'_{\ell, \text{row}}\|^2}{\|\Delta \mathbf{h}'_\ell\|^2} = \frac{\sum_{i=1}^r c_i^2}{\sum_{i=1}^H c_i^2}.$$

Results. Figure 6 confirms that update norms are nearly identical for easy and hard tokens throughout the network. Figure 5 shows that frac_{row} is close to 1 for both groups at every layer, ruling out null-space escape as an explanation for where the varying stabilization patterns in the vocab space appear in the hidden space. Figure 8 shows the decomposition of $\Delta \mathbf{h}'_{\ell, \text{row}}$ across bands of singular directions: for each band we sum the squared coefficients c_i^2 and normalize by $\|\Delta \mathbf{h}'_{\ell, \text{row}}\|^2$. Despite the outsized influence of $\sigma_1 \approx 776$ relative to the remaining singular values (Figure 7), the spectral profiles of easy and hard tokens are nearly identical for every layer transition.

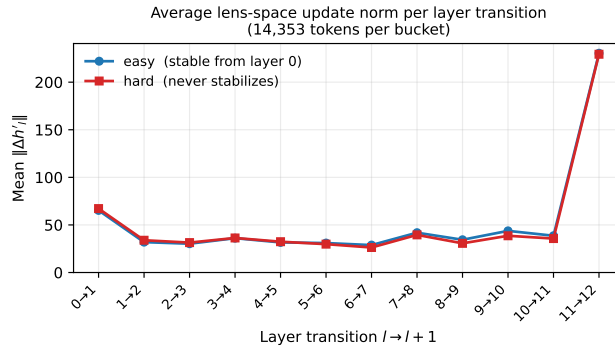


Figure 6: Norm of translated hidden state updates $\|\Delta \mathbf{h}'\|$, which is roughly the same for both easy and hard tokens across all layer transitions.

W_U SVD — shape 50304×768, rank 753

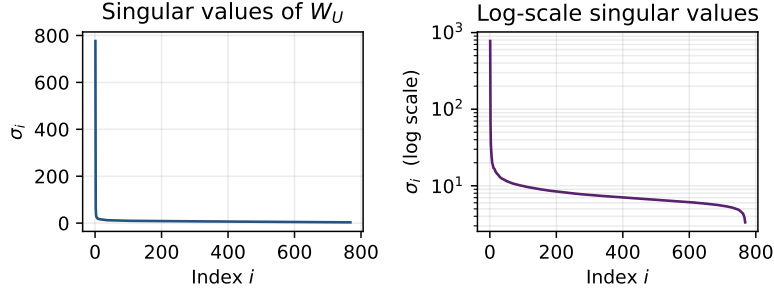


Figure 7: Singular values of the unembedding matrix W_U . $\sigma_1 \approx 776$ is much greater than all other singular values.

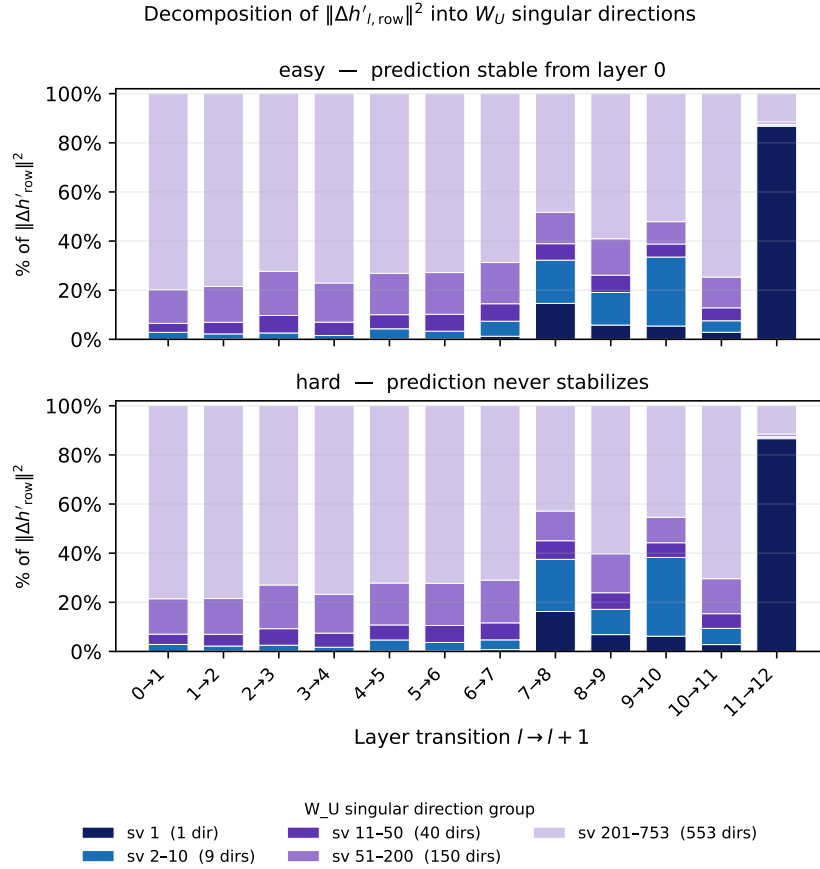


Figure 8: Decomposition of $\Delta h'_{\ell, \text{row}}$ across bands of singular directions: for each band we sum the squared coefficients c_i^2 and normalize by $\|\Delta h'_{\ell, \text{row}}\|^2$. The spectral profiles of easy and hard tokens are nearly identical for every layer transition.

8.2 Bootstrap uncertainty estimation

To quantify uncertainty in the affine map fittings, we use bootstrapping. For each token category, we generate $B = 200$ bootstrap replicates by sampling N tokens *with replacement* independently from the full pool of N tokens. The tokens that never get selected form the out-of-bag (OOB) set for that replicate.

For each replicate r , we fit a separate affine map $(W_{lr}, \mathbf{b}_{lr})$ for each layer transition $l \rightarrow l + 1$ on the in-bag tokens. We evaluate performance using the correlation coefficient, R^2 , and the mean squared error (MSE) on the OOB tokens, yielding $B = 200$ independent performance estimates per layer. We report the per-layer median as the central estimate and the 2.5th–97.5th percentile interval as a 95% confidence band, as shown in 3.

References

- [1] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-Depths: Dynamically allocating compute in transformer-based language models, April 2024. arXiv:2404.02258 [cs].
- [2] Aman Sharma, Saeed Najafi, Parsa Farinneya, Benyamin Jamialahmadi, Marzieh S. Tahaei, Yuhe Fan, Mehdi Rezagholizadeh, Boxing Chen, and Aref Jafari. DTRNet: Dynamic Token Routing Network to Reduce Quadratic Costs in Transformers, August 2025. arXiv:2509.00925 [cs].
- [3] Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting Latent Predictions from Transformers with the Tuned Lens, November 2025. arXiv:2303.08112 [cs].
- [4] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident Adaptive Language Modeling, October 2022. arXiv:2207.07061 [cs].
- [5] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A. Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, 2024. arXiv:2404.16710 [cs].
- [6] Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive Layer-skipping in Pre-trained LLMs. 2025.
- [7] Chen Zhang, Meizhi Zhong, Qimeng Wang, Xuanta Lu, Zheyu Ye, Chengqiang Lu, Yan Gao, Yao Hu, Kehai Chen, Min Zhang, and Dawei Song. MoDification: Mixture of Depths Made Easy, October 2024. arXiv:2410.14268 [cs].
- [8] Róbert Csordás, Christopher D. Manning, and Christopher Potts. Do Language Models Use Their Depth Efficiently?, October 2025. arXiv:2505.13898 [cs].
- [9] Akshat Gupta, Jay Yeung, Gopala Anumanchipalli, and Anna Ivanova. How Do LLMs Use Their Depth?, October 2025. arXiv:2510.18871 [cs] version: 1.
- [10] Stella Biderman, Hailey Schoelkopf, Quentin 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, May 2023. arXiv:2304.01373 [cs].
- [11] nostalgebraist. interpreting GPT: the logit lens — LessWrong. August 2020.