Exploring Monotonicity in Early-Exiting Language Models

Filipe Laitenberger [1] Max Belitsky [1] Denys Sheremet [1] Oliver Savolainen [1] Mark Bodracska [1]

Abstract

Large Language Models (LLMs) have shown impressive results across the board, but inference can be costly. A promising solution is posed by early exiting methods that assume that not all tokens need the same amount of computation, exiting the LLM at earlier layers. Several early exiting methods have been proposed, which rely on the implicit assumption that as the network does more computation, it will become more confident in its prediction. We investigate this assumption for two early exiting methods and propose three new confidence measures for early exiting based on the insights. We find early evidence for monotonicity benefitting the quality of token generation.

1. Introduction

Large Language Models (LLMs) are showing abilities that are beyond what was deemed possible even two or three years ago (Wei et al., 2022; Achiam et al., 2024; Anil et al., 2024; Touvron et al., 2023). These unprecedented results originate from ever-increasing model and dataset sizes which in turn lead to immense consumption of energy and resources, as well as environmental pollution (Strubell et al., 2019; Li et al., 2023; Patterson et al., 2021).

Because of this problem, research has been invested in making LLMs more efficient. One possible way of achieving this is adaptive computation allocation, which can be realized as a network solely utilizing certain sub-networks for specialized tasks (Jiang et al., 2024), skipping layers (Raposo et al., 2024) or early exiting a network (Panda et al., 2016; Teerapittayanon et al., 2016; Xin et al., 2021; Mangrulkar et al., 2022; Elbayad et al., 2019; Schuster et al., 2022; Bae et al., 2023; Geva et al., 2022; Del Corro et al., 2023; Elhoushi et al., 2024).

The current work focuses on early exiting in Transformers. Early exiting builds on the implicit assumption that the confidence of a model in its prediction will increase, the more computation it performs on a token, which we refer to as the *monotonicity assumption*. We (1) investigate the monotonicity assumption in prominent early exiting architectures (Schuster et al., 2022; Bae et al., 2023). We confirm that a weighted cross-entropy learning objective drives the model to decide on a prediction as early as possible, leading to mostly monotonic behavior after a certain layer. Furthermore, we (2) explore the hidden states of a network produced by processing sequences of different difficulty levels and examine the effects of the difficulty levels on hidden state saturation and monotonicity. Based on the findings, we (3) propose new confidence measures that exploit monotonic behavior and study their effect.

2. Related work

2.1. Early Exiting in Neural Networks

While neural networks are traditionally composed of many layers that sequentially process input tokens, early exiting assumes that not all inputs need the same amount of computation. Consequently, ”easy” token sequences could be output at earlier layers than ”difficult” ones, which need to traverse the entire network. Having been pioneered in CNN architectures (Panda et al., 2016; Teerapittayanon et al., 2016), early exiting has been studied in Transformers as well, including encoder (Xin et al., 2021; Mangrulkar et al., 2022), encoder-decoder (Elbayad et al., 2019; Schuster et al., 2022; Bae et al., 2023) and decoder models (Geva et al., 2022; Del Corro et al., 2023; Elhoushi et al., 2024). We specifically look at two works that aim to model the confidence or uncertainty of a model when generating tokens:

CALM. Schuster et al. (2022) fine-tune an LLM with a weighted cross-entropy objective that optimizes each layer to output the correct output probabilities given a shared LM-head:

$$\mathcal{L} = \sum_{i=1}^{L} \alpha_i \mathcal{L}_i \quad \text{where} \quad \alpha_i = \frac{i}{\sum_{j=1}^{L} j}$$

where $\mathcal{L}_i$ is the cross-entropy loss using each layer's hidden state, and α_i favors higher layers according to the equation above.

The authors further experiment with three different confi-

[*]Equal contribution [1]Department of Artificial Intelligence, University of Amsterdam, Amsterdam, Netherlands. Correspondence to: Filipe Laitenberger <filipe.laitenberger@student.uva.nl>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

dence measures: (1) computing the word probabilities from the current hidden state after each Transformer layer and exiting if the difference between the top two probabilities exceeds a calibrated threshold; (2) computing the cosine similarity between the current and last hidden state, and exiting if the similarity surpasses a calibrated threshold; (3) using a classifier that predicts the likelihood of early exiting based on the current hidden state.

Even though CALM aims to make token prediction faster by exiting early, a challenge that remains is handling attention between tokens when some have exited earlier than others, requiring individual copying of hidden states, which slows down the computation, especially as the number of layers increases (Bae et al., 2023).

FREE. Bae et al. (2023) extend CALM, trading compute adaptability for decreased overhead. Specifically, the authors reduce the number of exit points to two compared to every layer so that the model can either exit at, e.g., the fourth layer or use the entire network. Accordingly, FREE can copy missing hidden states in parallel to reduce overhead. Lastly, FREE replaces the expensively calibrated confidence thresholds used in CALM by learned ones. In addition to the weighted cross-entropy objective, FREE uses a layerwise knowledge distillation loss

$$\mathcal{L}_{\text{KD}} = \frac{1}{|L_S|} \sum_{i=1}^{L_S} \text{MSE}\left(\mathbf{H}_S^i, \mathbf{H}_D^{m(i)}\right)$$

where $\mathbf{H}_S^i$ refers to the hidden state in the shallow module, i.e., the hidden state after layer smaller than the total number of layers, and $\mathbf{H}_D^{m(i)}$ refers to the hidden state in the deep module, i.e., the hidden state after the full network pass. $m(i)$ either (1) maps to the last layer, (2) is a uniform mapping from shallow to deep layers, or (3) maps to the closest hidden state in the deep module, i.e., $m(i) = \arg\min_j \text{MSE}\left(\mathbf{H}_S^i, \mathbf{H}_D^j\right)$.

3. Do Early-Exiting Networks Behave Monotonically?

Early exiting only makes sense under the assumption that a model becomes more confident of a decision over time. Conversely, if a model exhibited purely unpredictable behavior throughout the evolution of hidden states across layers, there would be no notion of a ”saturated” hidden state that can be used for prediction since the prediction could always change completely after the current later. We performed extensive experiments to uncover whether this assumption holds, detailed in appendix A. We conclude that for the layerwise weighted cross-entropy used in CALM, the assumption holds, leading the model to decide on predictions as early as possible while showing consitently increasing confidence in the prediction.

In addition to that, we examine hidden state saturation, i.e., the development of hidden states across transformer layers, in greater detail in appendix B. Specifically, we visualize the saturation of the hidden states at each layer by using the cosine similarity between the current and eventual hidden state of CALM throughout the evaluation dataset, showing how the model handles sequences of varying difficulty. The results give additional evidence for monotonic behavior in this model.

4. Monotonic Early Exiting

Based on our observations above, we hypothesize that an exit mechanism could be improved in terms of performance on the metric of interest while retaining the benefits of early exiting if conditioned on multiple previous layers' hidden states. If the model is trained using the weighted cross-entropy objective, the monotonic patterns in the hidden states could inform the exit mechanism to be more confident of a decision based on the evolvement of hidden states rather than just the current hidden state. Henceforth, we develop, train, and test three new confidence measures:

LSTM-based classifier. A two-layered LSTM network (Hochreiter & Schmidhuber, 1997) with the input dimensionality equal to the transformer's hidden dimensionality, and two outputs representing exit and no-exit.

Classifier based on three previous hidden states. A two-layered MLP with three times the transformer's hidden dimensionality as inputs, ReLU activation (Agarap, 2018), the transformer's hidden dimensionality as hidden neurons, and two outputs representing exit and no-exit.

A heuristic method based on the last three top-1 softmax scores. The network makes an exit decision if and only if the last three layer's softmax scores are monotonically increasing and the current top-1 confidence is above 0.9. This handcrafted rule is based on the observations from our monotonicity experiments where three consecutively increasing top-1 scores above the named threshold would almost never change later on.

We hypothesize that the quality of the generation will be higher than CALM's confidence measures due to the increased quality of exit decisions and that latency will be higher due to increased computational complexity.

Comparison to CALM. With its hidden state saturation confidence measure, CALM experiments with one measure informed by the current and previous hidden state, similar to our method. Our endeavors into hidden state saturation reveal its frequent presence. However, there may be cases without hidden state saturation even though the model is

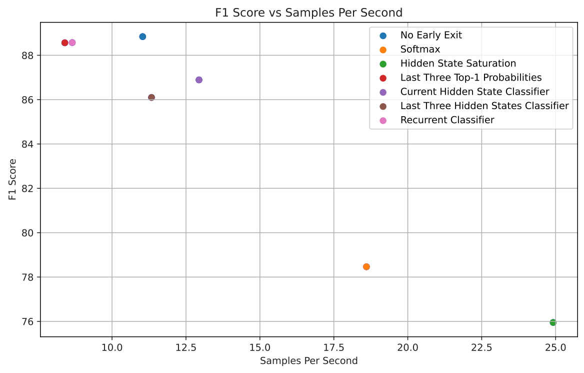

Figure 1. F1-score vs. samples per second on the SQuAD dataset.

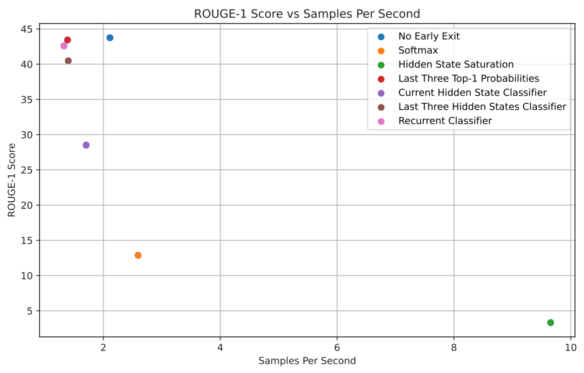

Figure 2. ROUGE-1-score vs. samples per second on the CNN-Dailymail dataset.

confident of a token, as very different hidden states can lead to the same softmax scores. This further motivates our classifier-based confidence measures conditioned on previous hidden states, which can recover the hidden state saturation behavior, and go beyond it if necessary. Compared to the original classifier used in the CALM paper, which didn't show great performance, our MLP takes in more hidden states as input which could give it useful information about how hidden states are changing and therefore perform better.

5. Experiments

Model. For all experiments, we use T5 models pre-trained on a layerwise weighted cross-entropy objective for monotonic behavior.

Datasets. We evaluate two datasets: (1) Open-book SQuAD 1.1 (Rajpurkar et al., 2016), a QA dataset out of Wikipedia articles complemented with questions and a target answer which is taken from the context article. (2) CNN/DM (Hermann et al., 2015), composed of news articles and target summarizations.

Baselines. We test the three confidence measures proposed by CALM: (1) The difference between top-1 and top-2 softmax score, (2) hidden-state saturation, and (3) a classifier trained on the current hidden state. Despite describing a framework for finding threshold values, the original CALM paper does not have details about final threshold values. Similar to Bae et al. (2023), we select 0.9 as the confidence threshold.

Novel Confidence Measures. We additionally test our three proposed confidence measures, comparing them to the three baselines. We use an identical loss function and training procedure proposed for the CALM classifier, training our classifiers for 5 epochs.

6. Results

The results for the SQuAD and CNN-Dailymail datasets are shown in Figure 1 and Figure 2 respectively. Two trends are observable throughout all datasets: (1) firstly, the quality of the generated tokens (measured in F1, and ROUGE-1 scores) increases substantially compared to the baseline confidence measures. (2) Secondly, our confidence measures mostly exhibit poor latency, being slower than all baselines (except for the last three hidden states classifier on the SQuAD dataset). These two trends give evidence that taking monotonicity into account greatly benefits the model's performance. However, the results also indicate that our confidence measures are poorly optimized for parallel computation or optimally frequent early exit decisions. A detailed interpretation is provided in Section 7.

7. Discussion

In this study, we conducted an in-depth examination of the monotonic behavior of early exiting models, initially formulating hypotheses and subsequently demonstrating how and under what conditions they exhibit increasing confidence in predictions over time. Based on these findings, we developed novel confidence measures leveraging this property, resulting in significantly higher scores compared to other confidence measures. Nevertheless, several questions remain that require further investigation:

Can the advantages of monotonicity be integrated while retaining low latency? Our methods show high quality of token generations, whereas CALM's latency seems to significantly benefit from considering only the current hidden state. While the current study focused on exploring whether monotonicity benefits early exiting, its continuation should give an overall picture of how fully optimized and calibrated architectures compare, answering whether monotonic early exiting can pose a new state-of-the-art.

Is early exiting the best way of making a model more efficient? We investigated whether early exiting depends on the monotonicity assumption, and that making the model decide as early as possible benefits the exit mechanism. Analogous to human thinking, we restrict the model to steer its thinking process in one direction quickly, which takes away the ability to ponder freely. However, model performance might benefit from the ability to explore many directions. There are two further directions we want to mention here:
(1) Prior work shows that LLMs use their first few layers to process the input without trying to make a final prediction (leading to somewhat random predictions in the first layers), while later layers are much more consistent in terms of the end prediction (Wendler et al., 2024). Thus, we hypothesize that early exit LMs may benefit from constraining the weighted cross-entropy objective to the last 70% layers of the model, restricting the model to a lesser extent and giving it more time to explore different directions at first.
(2) A different approach is Mixture-of-Depths (MoD) (Raposo et al., 2024) which skips layers instead of exiting altogether. This alleviates the model of having to be monotonic and hence does not restrict it at all. The model can thus exhibit completely random behavior, process tokens in many different ways, and simultaneously decide to skip certain parts, specializing different stages for different processing steps of a token. MoD may be combined with a weighted cross-entropy objective in the later layers, so that the first part of the model can ponder freely while retaining the ability to skip layers, and the second part of the network can decide when a token can exit altogether.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., (...), J. B., and Zoph, B. Gpt-4 technical report, 2024.

Agarap, A. F. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL `http://arxiv.org/abs/1803.08375`.

Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., (...), O. F., and Vinyals, O. Gemini: A family of highly capable multimodal models, 2024.

Bae, S., Ko, J., Song, H., and Yun, S.-Y. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. *arXiv preprint arXiv:2310.05424*, 2023.

Del Corro, L., Del Giorno, A., Agarwal, S., Yu, B., Awadallah, A., and Mukherjee, S. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*, 2023.

Elbayad, M., Gu, J., Grave, E., and Auli, M. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Geva, M., Caciularu, A., Wang, K. R., and Goldberg, Y. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts, 2024.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, P., Yang, J., Islam, M. A., and Ren, S. Making ai less” thirsty”: Uncovering and addressing the secret water footprint of ai models. *arXiv preprint arXiv:2304.03271*, 2023.

Mangrulkar, S., MS, A., and Sembium, V. Be3r: Bert based early-exit using expert routing. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3504–3512, 2022.

Panda, P., Sengupta, A., and Roy, K. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 475–480. IEEE, 2016.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Raposo, D., Ritter, S., Richards, B., Lillicrap, T., Humphreys, P. C., and Santoro, A. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V., Tay, Y., and Metzler, D. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

See, A., Liu, P. J., and Manning, C. D. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099.

Sharma, E., Li, C., and Wang, L. Bigpatent: A large-scale dataset for abstractive and coherent summarization. *arXiv preprint arXiv:1906.03741*, 2019.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wendler, C., Veselovsky, V., Monea, G., and West, R. Do llamas work in english? on the latent language of multilingual transformers, 2024.

Xin, J., Tang, R., Yu, Y., and Lin, J. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pp. 91–104, 2021.

A. Do Early-Exiting Networks Behave Monotonically?

All the methods discussed above assume that confidence evolves monotonically, i.e., that the model will be more certain of a prediction the more computation it performs on a token. This assumption is central to the functioning of early exit methods regarding the decision when to exit and whether it is sensible to exit early in the first place - it could be that the evolvement of hidden states is utterly unpredictable and does not resemble any meaningful connection to the eventual word probabilities at the final layer, i.e., the network might be a black box whose intermediate representations are meaningful to itself but not to the outside world. On the other hand, it might be that intermediate hidden states can be seen as "contemplation" of the model, or that the model even tries to decide on a prediction as early as possible in its contemplation process.

Experiment To test whether this monotonicity assumption holds, we conduct an experiment on three different settings of a T5 model - a default variant without early exiting, a CALM model optimized on the weighted cross-entropy objective, and a FREE model that uses the additional layerwise knowledge distillation term. For each model, we use publicly available pre-trained checkpoints and evaluate each model on summarization using the BigPatent dataset (Sharma et al., 2019). Specifically, we leverage the models' LM-heads after each of the twelve layers to compute two things: (1) the fraction of tokens for which the prediction does not change after the respective layer, i.e., whether the model could have exited early at the respective layer without a loss in the performance metric of interest. (2) A plot showing the mean and standard deviation of the confidence in the eventual prediction at each layer. (3) Plots showing the top-3 predictions at each layer for individual token generations.

Results (1) Figure 3 shows that at the second layer already, the weighted cross entropy objective optimizes the model to be confident of a token, i.e. not change its top-1 prediction after layer two, in 75% of the cases. Additionally, after the fourth layer, the model keeps its prediction in 90% of the cases. Contrastingly, the default T5 and even FREE exhibit much less certainty in their predictions. Even though FREE uses the weighted cross-entropy objective as well, its additional layerwise distillation objective seems to be inhibiting the same monotonic behavior as in CALM.
(2) In addition, Figure 4 demonstrates that CALM increases monotonically in its top-1 prediction at early layers on average. Meanwhile, the vanilla T5 gains confidence much later. On the other hand, FREE displays locally monotonic behavior, i.e., its confidence increases until its first exit point, then drops and increases again until the end of the network.

(3) To illustrate this behavior in individual cases, figure **??** - 7 depicts three example forward passes of the same

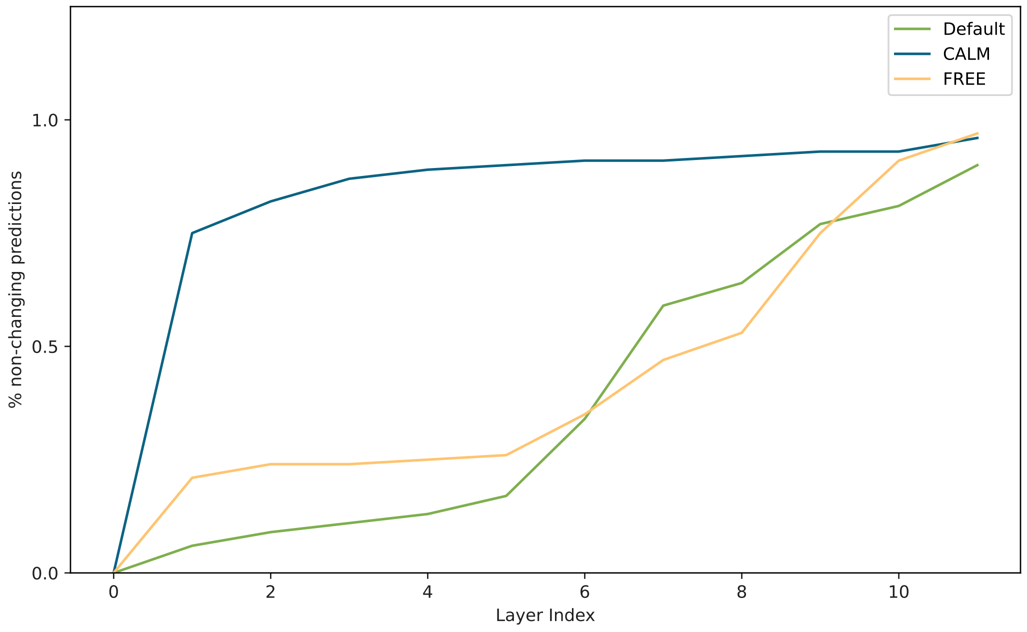

Figure 3. The fraction of top-1 predictions that do not change after each layer, measured across the BigPatent evaluation dataset.

sequence for the three models, with the default T5 on the left, CALM in the middle, and FREE on the right. The plots exemplify the rather unpredictable behavior of the default and FREE models, while CALM decides on a prediction as early as possible. While CALM's confidence increase shows slightly non-monotonic evolvement in earlier layers, the plots exemplify its monotonicity in later layers.

Conclusion We show that the weighted cross-entropy objective encourages the model to decide on a prediction as early as possible while exhibiting monotonically increasing behavior in its predictions. These results indicate that an early exit mechanism could benefit from taking this behavior into account, exiting early based on whether it observes monotonically increasing predictions.

B. Sequence types: which sequences are ”easy” and which are ”difficult”?

In the previous section, we have seen that a Transformer model trained with a weighted cross-entropy objective exhibits a monotonic pattern in token predictions. However, even having this property does not make the model confident to exit early on every possible sequence. Naturally, some sequences in a language are more ambiguous than others. For instance, the sequence *One of the biggest cities in the world is New* _ can be considered ”easy” as the next word is most likely to be *York* due to this being factual knowledge. On the other hand, the sequence *The students went to* _ is not that easy to predict even for a human being, as it bears an inherent degree of uncertainty without having any context.

In this section we look at the properties of the hidden states of such a monotonic network and what they can tell us about the difficulty of the input sequences.

Experiment In order to identify which sequences are ”difficult” or ”easy” for a model we conduct the following experiment: we select 2500 examples from the validation set of the CNN Daily Mail summarization dataset (See et al., 2017), iteratively feed the sequences to the model in an autoregressive manner and record the hidden states of each sequence. This procedure produces n sequences. Since we use T5 large, we get 24 hidden states for each sequence. We then compute the cosine similarity between the last hidden state (which is used for the next token prediction) and hidden states after each other layer. The resulting vector shows how saturated the hidden states of an input sequence are. If the hidden states saturate quickly (become similar to the last hidden state after 4-6 layers), the benefits of further computations can be considered small. That means that the network can exit earlier on such sequences. We deem the sequences on which the network exhibits the described behavior ”easy” sequences. On the other hand, the sequences that require almost a full pass through the network for the hidden states to saturate are deemed ”difficult”.

We compute the mean of the hidden state similarities for each sequence of tokens, sort the means in descending order, and select the first 1000 items to obtain the ”easy” sequences. The same procedure but in ascending order is repeated to obtain the ”difficult” sequences. Additionally, to investigate

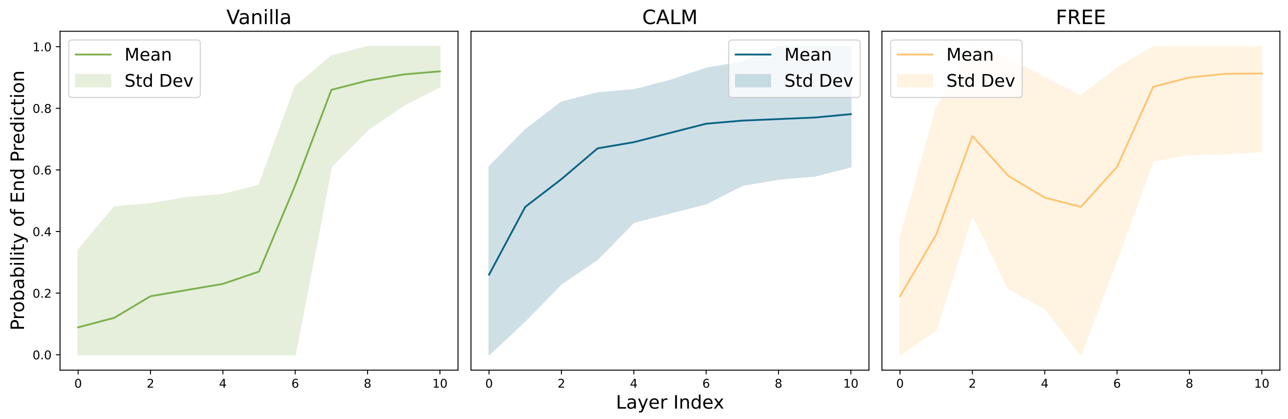

Figure 4. The mean and standard deviation of the end prediction across the BigPatent evaluation dataset, with the default T5 on the left, CALM in the middle, and FREE on the right.

Table 1. The properties of sequence types.

	Easy	**Difficult**
First similar state	8.65 (1.34)	18.46 (0.76)
Similar states	14.35 (1.34)	4.54 (0.76)
Sequence length	58.16 (31.81)	11.48 (20.69)
Monotonic (all)	24%	3%
Monotonic (after l_4)	78%	97%
Monotonic (after l_8)	85%	98%

the properties of these sequences, we compute the following metrics: the index of the first layer with a saturated hidden state, the number of similar hidden states, the sequence length, and a boolean variable that indicates whether the hidden state similarities are strictly increasing after layers 0, 4 and 8. The last metrics were added to further investigate the monotonicity property and aid in coming up with better confidence measures. The similarity threshold was set at 0.9.

Results Table 1 shows the results of the above experiment. The first finding is that (1) the mean layer index of the first similar state for ”easy” sequences is 8.65, whereas for the ”difficult” sequences that number is 18.46, which is close to the total number of layers in the model (24). In addition to that, the average number of similar hidden states in ”easy” sequences is much larger than in ”difficult” sequences with 14.35 and 4.54 hidden states respectively. This suggests that the hidden states of the ”easy” sequences saturate much faster and do not require the full pass through the model, whereas the ”difficult” sequences tend to saturate closer to the last layer.

Another significant observation is that (2) ”easy” sequences tend to be considerably longer than ”difficult” sequences, with 58.16 and 11.48 tokens on average respectively. This phenomenon is logical, as the space of potential tokens that can be generated is substantially larger at the beginning of the sequence generation process. In contrast, with longer sequences, the model is able to leverage contextual information, thereby making the distribution over vocabulary sharper, narrowing the scope of possible tokens.

Finally, the results on the monotonicity of the hidden state similarities indicate that 24% of ”easy” sequences are strictly increasing from the layer, which is not the case for ”difficult” sequences. However, the ”difficult” sequences do exhibit monotonic behavior later in the network: 97% are monotonic after 4 layers and 98% are monotonic after 8 layers.

In addition to the quantitative analysis, we have plotted the hidden state similarities to inspect the differences visually. Appendix C shows examples of how the hidden states evolve over layers on some examples of sequences of both types.

Conclusion These results indicate that an early exit mechanism could benefit from taking the sequence length into account. Additionally, it can be noted that even for the ”easiest” sequences the first few layers of the network cannot be used for early exiting, which should be accounted for by the early exiting mechanism as well to not waste computation on confidence measures for these layers.

C. Sequence type examples

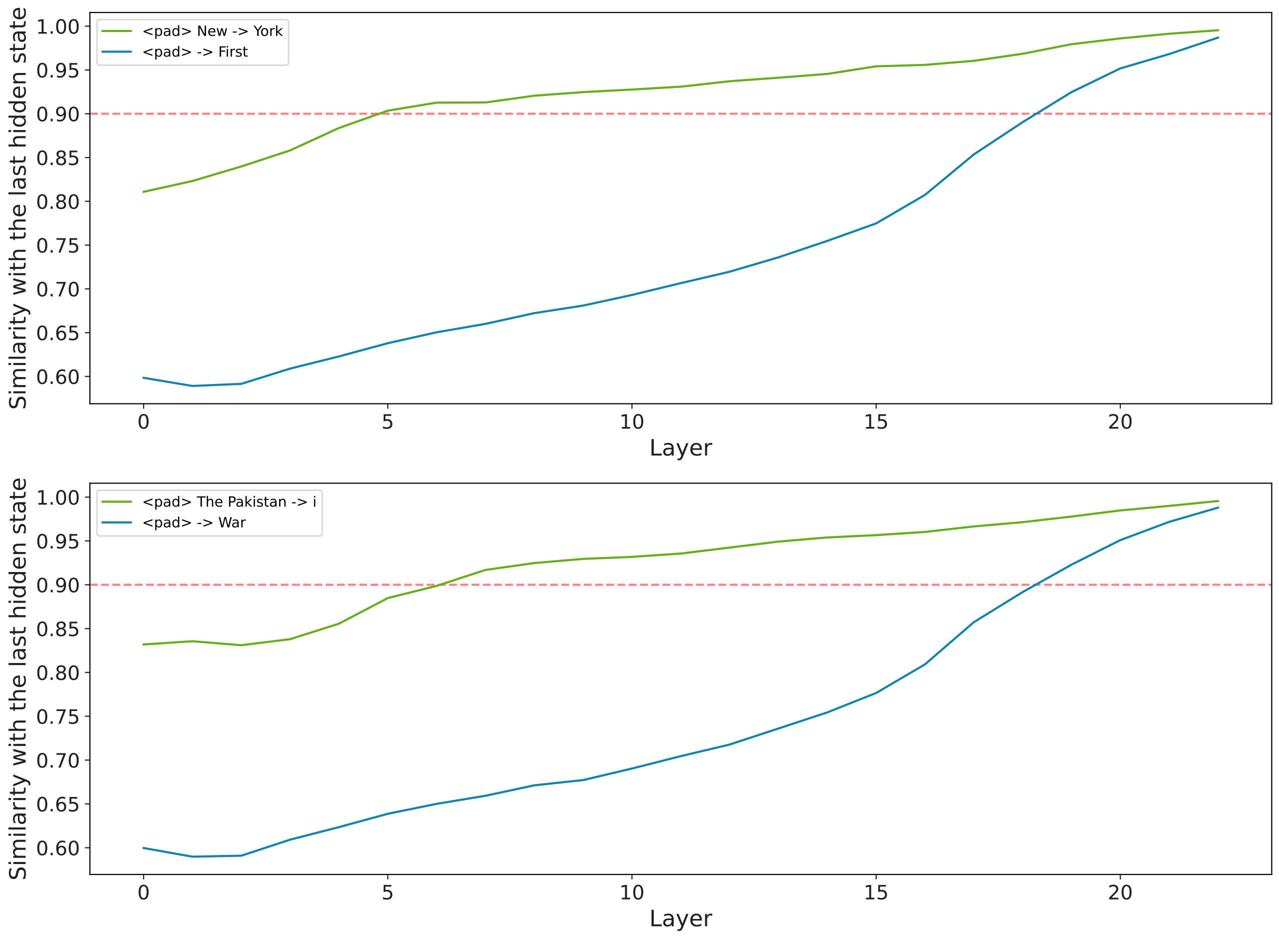

Figure 6. Similarities between the last and every other hidden state. The green line corresponds to the ”easy” sequence and the blue line corresponds to the ”difficult” sequence.

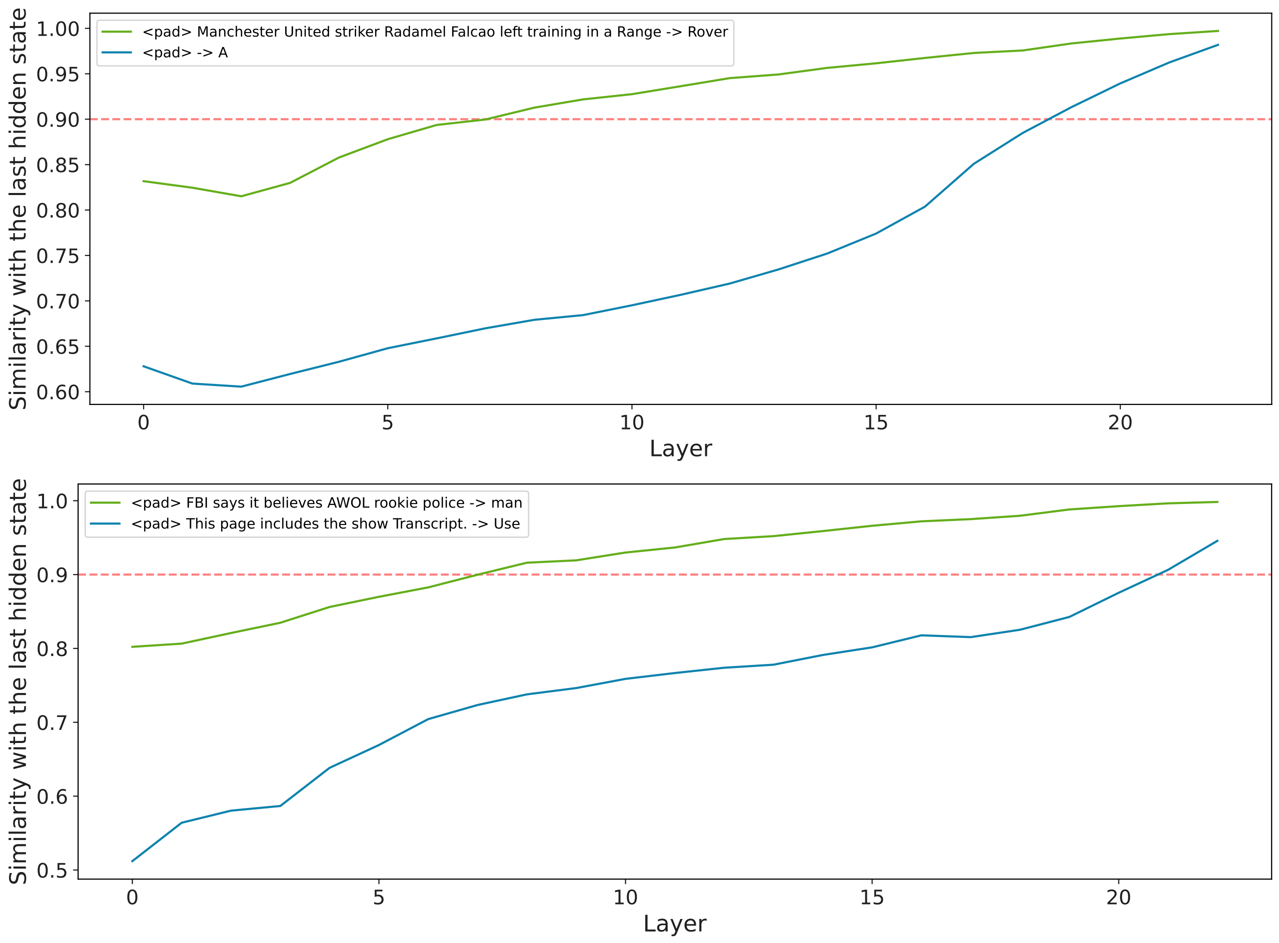

Figure 7. Similarities between the last and every other hidden state. The green line corresponds to the ”easy” sequence and the blue line corresponds to the ”difficult” sequence.