Revisiting Funnel Transformers for Modern LLM Architectures with Comprehensive Ablations in Training and Inference Configurations

Anonymous ACL submission

Abstract

Transformer-based Large Language Models (LLMs) suffer from high computational costs, especially as the runtime scales quadratically with sequence length. However, architectures advance so quickly that techniques proposed to streamline earlier iterations are not guaranteed to benefit more modern models. Building upon the Funnel Transformer proposed by Dai and Le (2020), which progressively compresses intermediate representations, we investigate the impact of funneling in contemporary Gemma Transformer architectures. We systematically evaluate various funnel configurations and recovery methods, comparing: (1) standard pretraining to funnel-aware pretraining strategies, (2) the impact of several funnel-aware fine-tuning configurations, and (3) different methods of sequence recovery. Our results demonstrate that funneling creates information bottlenecks that propagate through deeper network layers, particularly in larger models (e.g., Gemma 7B), leading to at times unmanageable performance lost. However, carefully selecting the funneling layer and employing effective recovery strategies can substantially mitigate performance losses, achieving up to a 44% reduction in latency. Our findings highlight key trade-offs between computational efficiency and model accuracy, providing practical guidance for deploying funnel-based approaches in large-scale natural language applications.

1 Introduction

In recent years, Transformer architectures have revolutionized natural language processing (NLP) by enabling models to capture complex dependencies within sequences. However, this capability comes at a significant computational cost, particularly when processing lengthy sequences, as the self-attention mechanism scales quadratically with sequence length. This scaling issue poses challenges for deploying large language models (LLMs) at production scales in real-world applications.

Building on the foundational work by Dai et al. (2020), who proposed the Funnel Transformer to progressively reduce sequence length by pooling intermediate representations, we extend their approach to modern transformer architectures. Dai et al. (2020) demonstrated their concept using the BERT architecture, which had groundbreaking impact for it's time. Although BERT has largely become antiquated compared to current state-of-the-art transformer-based models, it remains implemented in industry (Gardazi et al., 2025), and so researching the upcycling of modern LLMs into encoder optimizations may have large impact.

In this work, we extend the Funnel Transformer concept to contemporary architectures by specifically testing modern Gemma models (Team et al., 2024). Unlike BERT, the Gemma family large dense models that may be used as encoders or decoders, and it is far from given that they should respond similarly to the impact of funneling. While the open source family of Google foundation models transitioned to decoder-only architectures, **we demonstrate that it is possible to distill them into task-specific encoders.** We make **a further leap in both by distilling them into funnel transformers**, which we show to perform even better in cost and task performance.

We further distinguish our study through a detailed experimental design that systematically evaluates key aspects of funneling. Specifically, we perform rigorous ablations to determine:

1. At which layer to pool intermediate representations (while targeting a maximum acceptable performance degradation of approximately 5%)

2. Which heuristic operation to use when recovering the full sequence length.

3. We investigate three training scenarios: pretraining transformers with funneling integrated, fine-tuning pretrained transformers exclusively with funneling, and performing inference with funnel transformers without prior funnel-aware pretraining or fine-tuning.

Through these comprehensive evaluations, we aim to quantify the performance trade-offs introduced by funneling and clearly delineate the circumstances under which funneling yields significant computational benefits while maintaining acceptable performance degradation. Our findings provide guidance on effectively leveraging funneling to balance efficiency gains with minimal performance sacrifices in practical, large-scale LLM deployments.

2 Background Literature

The high computational costs of Transformer-based large language models (LLMs) motivate research into more efficient architectures (Tay et al., 2022; Jin et al., 2024). This research aims to reduce costs like FLOPs, memory, and energy while maintaining model accuracy. Innovations include sparse Mixture-of-Experts (MoE) models (Shazeer et al., 2017; Fedus et al., 2022; Du et al., 2022; Lu et al., 2024; Zuo et al., 2022) and new model structures.

Despite the focus on decoder-only models, encoder architectures offer an effective approach to efficiency in certain contexts. Encoder-only Transformers, like BERT (Devlin et al., 2019), process inputs bidirectionally, outputting sequence or pooled representations. They can be more efficient than decoders for specific industrial tasks requiring accuracy, low cost, and scalability. For example, new encoders like ModernBERT improve on quality and cost (Warner et al., 2024). Interest is also growing in upcycling decoder models into encoders, like Dec2Enc, which adapts models such as Qwen 2.5 and Gemma-2 (Huang et al., 2025).

The Funnel-Transformer (Dai et al., 2020) is an example of an efficient encoder. It uses a downsampling encoder to gradually compress token sequences into shorter hidden representations. The rationale is that not all tokens are needed for high-level understanding, especially for tasks like classification. Funneling reduces hidden states in deeper layers, cutting self-attention and feed-forward network (FFN) costs. If needed, a lightweight decoder can up-sample the representation for token-level outputs, similar to an auto-encoder. However, funneling approaches have largely been ignored after its initial introduction in 2020.

Funnel models are well-suited for hybrid architectures using retrieval or external knowledge, as seen in systems like FunnelRAG (Zhao et al., 2025), RaSeRec (Zhao et al., 2024), and Condenser (Gao and Callan, 2021). For instance, a retriever can fetch documents, which a funnel Transformer compresses into a vector for a decoder to generate an answer. This two-stage system is typically faster and more memory-efficient than large end-to-end models.

A recurring theme in the field of LLM efficiency is sparsity: not all parts of a model or input need uniform processing. Architectures like funnel models, quantization (Lang et al., 2024), and pruning (Cheng et al., 2024) manage computational resources by focusing on essential information, leading to more scalable and accessible LLMs.

3 Model Design

In our experiments, we perform all training on Gemma 2b and 7b models[1] We used the Gemma 1 checkpoint to simplify the implementation of the pooling mechanism by not considering the alternating sliding window attention layers in Gemma 2 and 3. This is not a fundamental limitation of pooling, but adds another dimension to tuning which we did not consider at the time of this writing.

We conduct a limited grid search and produce a hyperparameter configuration that we keep constant across model architectures, listed in the Appendix. Please find the hyperparameters for Gemma 2B at Table 2 and the hyperparameters for Gemma 7B at Table 3.

We perform all of our experiments on a TPU-based computational setup. Our computational budget was on the order of O(1000) TPU hours.

4 Experimental Setup

4.1 Benchmarks studied

In this work, we primarily report results on the **General Language Understanding Evaluation (GLUE)** benchmark and the **CoNLL-2003 Named Entity Recognition (NER)**. **GLUE** is a collection of nine natural language understanding

[1]The Gemma license from Google fosters broad use and innovation for its open language models, permitting commercial applications, redistribution, and modifications, alongside a focus on responsible AI development (Team et al., 2024).

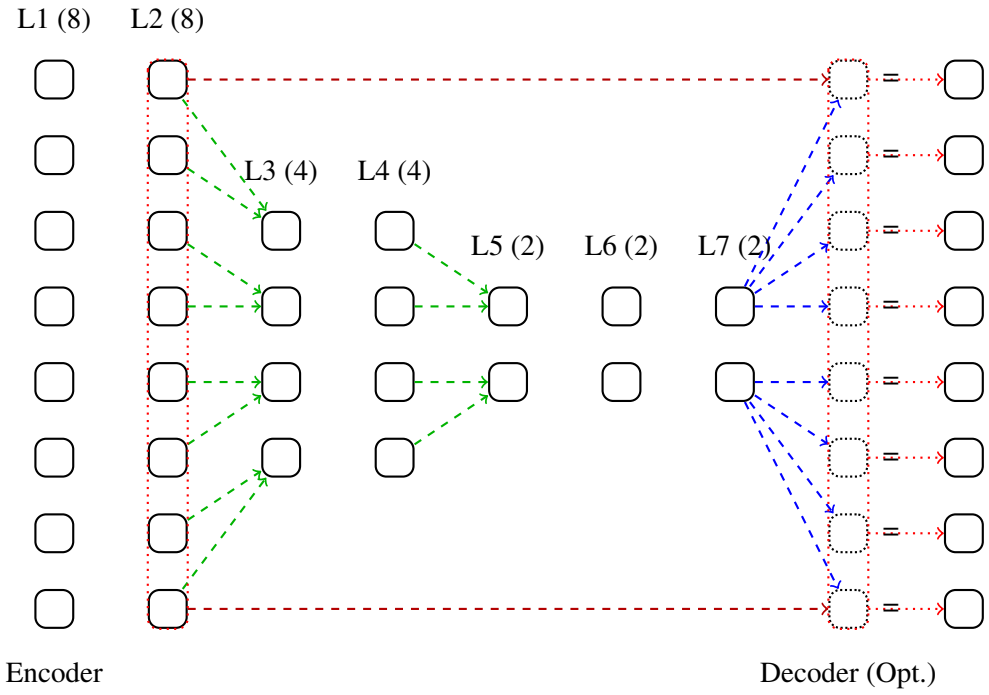

Figure 1: A representation of a funnel architecture with seven encoder layers. Funneling operations reduce blocks from 8 to 4 between L2 and L3, and from 4 to 2 between L4 and L5. Skip connections and up-sampling to an optional decoder are shown. (Layers are 1-indexed for this caption.)

tasks designed to evaluate and analyze the performance of models across diverse linguistic phenomena. Introduced by (GLU), GLUE includes tasks such as single-sentence classification, similarity and paraphrase detection, and natural language inference, providing a comprehensive platform for assessing general language understanding capabilities. We include GLUE as a *sentence level classification* task [2].

The **CoNLL-2003 Named Entity Recognition (NER)** (Sang and Meulder, 2003) benchmark is a widely utilized dataset for evaluating NER systems in English. It consists of text annotated with four entity types: persons, locations, organizations, and miscellaneous entities. Models are assessed using the F1 score, which balances precision and recall. We present the average scores between the English and German NER. We include NER as a *token-level classification* task.

In addition, we report results on an internal, carefully curated benchmark called **WebAnswers Classification**. In order to maintain confidentiality and anonymity, we opt not to describe the benchmark in this reviewing phase. The metric that we report from this benchmark is the ROC AUC of the classification task. WebAnswers, like GLUE, is *sentence-level* classification.

[2]It is important to note that our reported GLUE scores are averaged over 8 to 9 runs with different random seeds, which displayed almost no variation between seeds – the performance was quite stable and repeatable. While additional plots zooming in to show the variance across different seeds could further illustrate these effects, the averaged scores already provide a reliable measure of performance consistency.

4.2 Testing the effect of pretraining on Funnel transformer Setup

We compare whether pretraining the transformer in a funnel-aware setup positively impacts the scores. In all cases, we pretrain the Gemma models for 100k steps on the standard Gemma pretraining corpus (Team et al., 2024); in the funnel-aware pretraining setup, we perform a max-pool two-token funnel once at layer 2 and then continue with the reduced dimension through the rest of the transformer. Once pretrained, we test the results of applying a max-pool two-token funnel to every even layer in a 16 layer Gemma model: that is, layers 2, 4, 6, 8, ..., 16. Please note that when the max-pool funnel is applied to the second layer, it matches the funnel-aware pretraining configuration exactly. We report results on sentence level tasks only, and thus we do not recover the sequence length in any of these experiments.

4.3 Fine-tuning a funnel-aware setup

We fine-tune the funnel-aware Gemma 2b and 7b models on all GLUE benchmarks and present an average. We fine-tune for 2000, 4000, and 6000 steps and perform a two-token max-pool funnel at every even layer of a 16-layer 2B Gemma model, and similarly at every even layer of the 28-layer 7B Gemma model. Here, we profile latency on the Gemma 2B models by measuring wall-clock completion across the benchmark. Similarly to the pretraining experiments, we report results on sentence level classification tasks only, and also hold constant the lack of sequence length recovery.

4.4 Optimizing the type of recovery operation

We ablate among different types of operations to recover the full sequence length *at the last layer* of the Gemma 2B model. In all variants, we tile the intermediate activations of the last funnel layer to recover the full dimension (that is, if the layer is half the original dimension due to funneling, we repeat each activation twice to create a layer of the original dimension; e.g. $(1, 3, 4) \rightarrow (1, 1, 3, 3, 4, 4)$. We test the following variants:

1. **Sum with first layer (baseline)**: similar to the operation used in (Dai et al., 2020), we add the tiled funnel activation to the output activations of the transformer's first layer.

2. **Sum with last layer**: we add the tiled funnel activation to the output activations of the final full layer before funneling was performed.

3. **Sum with all the previous layers' max**: we add the tiled funnel activation to the maximum index-wise activation of all prior layers.

4. **Sum with all the previous layers' avg**: we add the tiled funnel activation to the average of the activations of all prior layers.

5. **Average with last layer**: we average the tiled funnel activation with the activation of the final pre-funnelled layer.

6. **Max(Last, Now)**: we take the index-wise maximum of the last pre-funnelled layer and the tiled funnel activation.

We compute performance across even steps of a two-token max-pool funnel that is applied throughout the 16 layers of the architecture. We present scores on the NER benchmark, as it is a token level task and thus relevant to recover the full sequence length.

5 Results

5.1 Effect of pretraining

Figure 2 displays two performance metrics as a function of the funnel recovery layer. The left panel shows the average GLUE score, while the right panel shows ROC AUC on the WebAnswers dataset. In each plot, the x-axis includes a 0 point that represents the "No Funnel" case. For the GLUE benchmark, the normal pretraining curve has a value of 88.81 at x=0, whereas the funnel-aware pretraining curve starts at 87.17. Similarly, in the WebAnswers plot the "No Funnel" baseline is 73.40 for normal pretraining and 72.85 for funnel-aware pretraining. For both metrics, performance is tracked over increasing funnel recovery layers (2, 4, . . . , 16).

5.2 Finetuning a funnel-aware setup

Please see Figure 3 for a depiction of the effects of fine-tuning funnel aware architectures across Gemma 2b and 7b models. More fine-tuning seems to help models perform better.

Figure 3 presents a set of four subplots comparing the performance of Gemma 2B and Gemma 7B models on two tasks: the GLUE benchmark (top row) and the WebAnswers ROC AUC (bottom row). In each subplot, the x-axis denotes the funnel recovery layer, with an x=0 point corresponding to the "No Funnel" case. Two performance curves are shown in each panel: one for models trained without funnel-aware pretraining (normal pretraining) and one with funnel-aware pretraining.

In the GLUE subplots (top row), the normal pretraining curve starts at an 88.81 score for Gemma 2B and 87.17 for Gemma 7B at x=0, while the funnel-aware pretraining curves start at 87.17 and lower, respectively. For the WebAnswers subplots (bottom row), the "No Funnel" baseline is 73.40 for normal pretraining and 72.85 for funnel-aware pretraining, with performance measured as WebAnswers ROC AUC. Benchmark metrics is plotted for increasing funnel recovery layers (2, 4, 6, . . . , 16 for Gemma 2B and 2, 4, . . . , 26 for Gemma 7B).

While the baseline of Gemma 7b is higher than the baseline of Gemma 2b, the performance of Gemma 7b seems to suffer more overall than the performance of Gemma 2b.

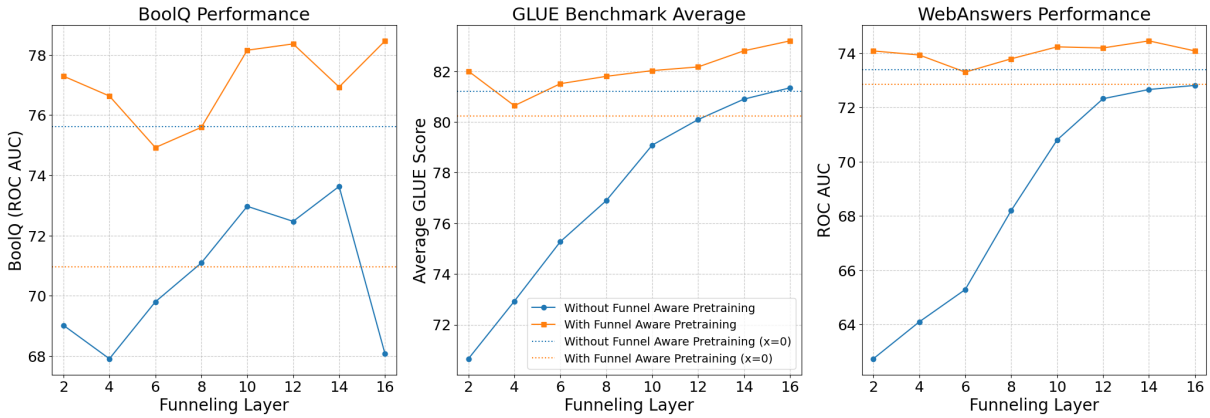

Figure 2: Performance on (a) BoolQ benchmark, (b) GLUE benchmark (Average GLUE Score) and (c) WebAnswers ROC AUC as a function of the funnel recovery layer. The x=0 point corresponds to the model without funneling. Solid lines represent models trained with normal pretraining ("Without Funnel Aware Pretraining") and funnel-aware pretraining ("With Funnel Aware Pretraining").

Table 1: Performance metrics for different funnel configurations. The left-most column, "Funnel Layer", corresponds with the x-axis of our charts and indicates at which layer the two-token max-pool funnel is applied.

Funnel Layer	stsb (Spearman)	cola (Acc.)	qqp (Acc.)	qnli (Acc.)	sst2 (Acc.)	rte (Acc.)	mrpc (Acc.)	mnli (m-Acc.)	mnli (mm-Acc.)	GLUE Avg.
normal pretraining										
2	59.79	66.73	83.89	78.56	84.98	53.43	70.83	68.65	69.03	**70.65**
4	67.82	67.21	84.00	80.62	86.70	55.60	66.42	73.08	74.89	**72.93**
6	77.47	67.11	85.11	82.94	85.67	55.96	72.30	75.13	75.77	**75.27**
8	84.51	65.77	86.79	84.02	88.07	55.60	72.79	76.72	77.78	**76.89**
10	87.02	69.61	86.90	86.34	88.88	56.68	78.19	78.95	79.19	**79.08**
12	86.95	75.55	86.84	86.40	89.56	59.57	78.43	78.66	78.89	**80.09**
14	87.64	76.99	86.54	86.75	89.33	59.57	81.37	79.74	80.19	**80.90**
16	87.83	76.70	86.53	87.92	89.56	59.57	82.35	80.66	81.01	**81.35**
funnel aware pretraining										
2	88.00	76.80	88.17	88.28	90.25	59.57	85.78	80.22	80.98	**82.01**
4	86.91	74.59	87.59	87.50	90.37	59.93	79.17	79.60	80.12	**80.64**
6	87.11	75.26	87.37	87.31	90.37	60.65	84.56	80.18	80.78	**81.51**
8	87.67	76.51	87.29	88.39	90.83	58.48	86.52	79.77	80.75	**81.80**
10	88.78	75.65	87.75	87.81	89.79	61.37	85.54	80.44	81.11	**82.03**
12	88.40	76.03	87.35	88.12	90.94	61.37	85.29	80.55	81.50	**82.17**
14	88.55	76.03	87.73	87.86	90.83	67.51	84.80	80.70	81.28	**82.81**
16	88.35	78.04	87.99	88.45	90.83	66.43	87.01	80.55	81.15	**83.20**

As shown in our latency results, i.e. Figure 4 plot, introducing the funneling layer at earlier stages provides substantial latency savings, with a peak of over 40% when funneling is applied at or near layer 0. However, as the funneling layer increases (moving further into the model), the latency gains steadily diminish. By layer 16, the latency savings drop to around 5%. This trend suggests that while early funneling can significantly speed up inference, its benefits taper off if the funneling is applied deeper in the network.

5.3 Optimizing the type of recovery operation

Please see Figure 5 for a depiction of the effects of different recovery operations on a Gemma 2b model.

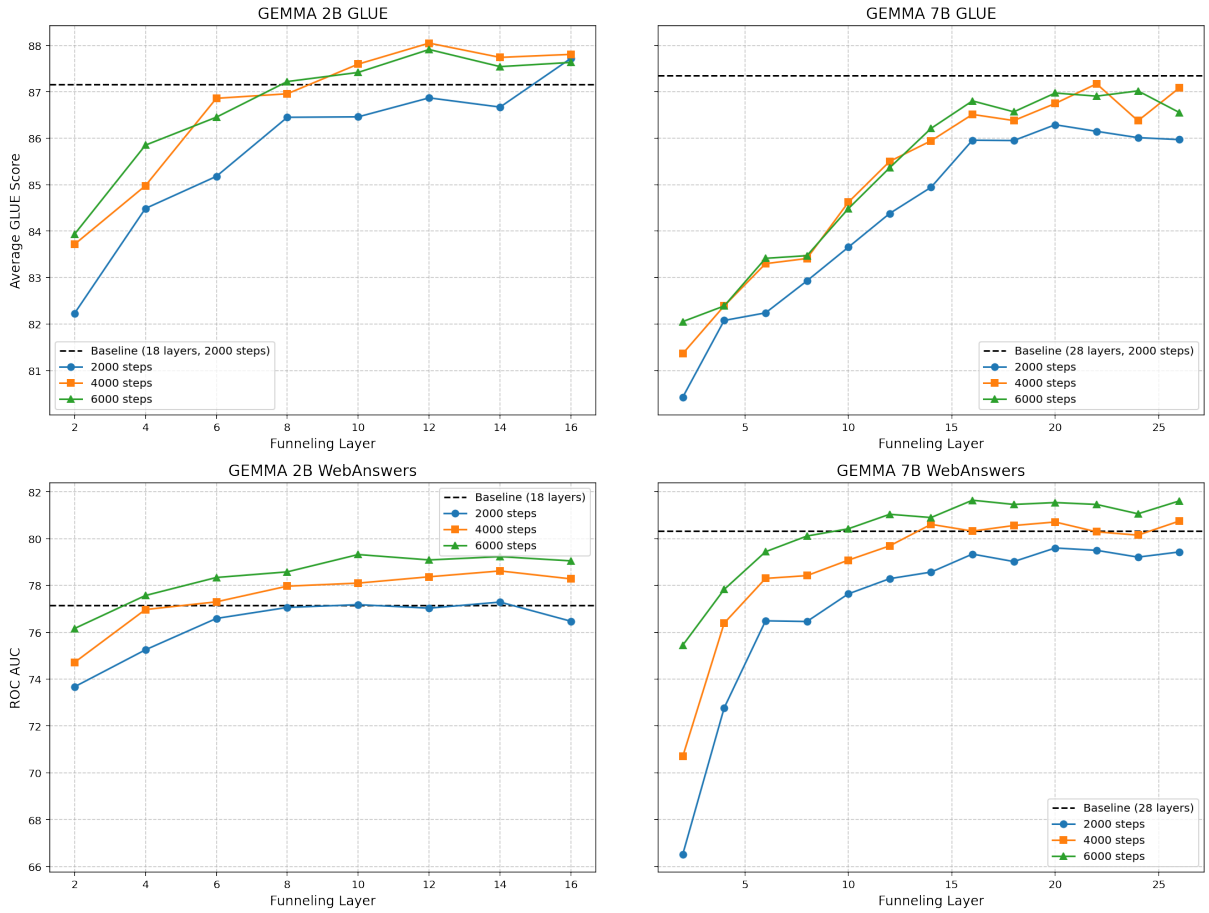

Figure 3: Performance of Gemma 2B (left) and Gemma 7B (right) on GLUE (top row) and WebAnswers ROC AUC (bottom row) are plotted against successive layers at which 2-token funnel is applied within each architecture. Solid lines correspond to models performance with different numbers of funnel-aware finetuning steps, whereas dotted lines correspond to baselines in which no funneling is applied.

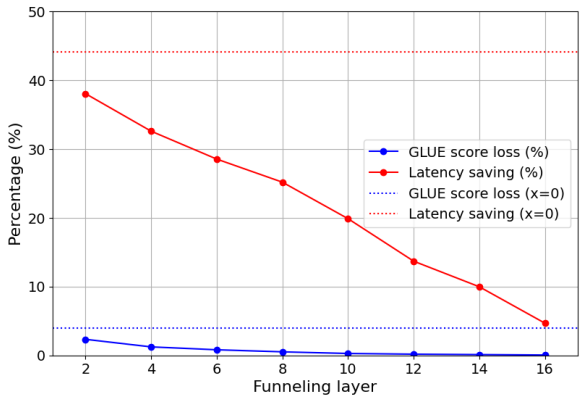

Figure 4: Comparison of latency versus performance gains.

6 Discussion

6.1 Impact of Pretraining on Accuracy and Quality Drop

Our results indicate that the model's accuracy follows a V-curve, where performance initially drops as the funnel configuration diverges from the original pretraining configuration. One plausible explanation is that pretraining helps to blunt this drop in quality. Specifically, the pretraining procedure incorporates a second layer that is funnel-aware, which appears to counterbalance the information loss introduced by the funnel configuration at that particular layer.

Interestingly, when no funneling is applied (x=0), models trained with normal pretraining outperform those with funnel-aware pretraining, causing the corresponding performance curves to cross over. This suggests that in the absence of any funnel-induced modifications, the additional complexity introduced by funnel-aware pretraining does not confer a benefit and may even be detrimental.

The shapes of the performance curves in both the GLUE and WebAnswers plots exhibit the characteristic V-pattern, with an initial decline in performance followed by a recovery as the funnel config-

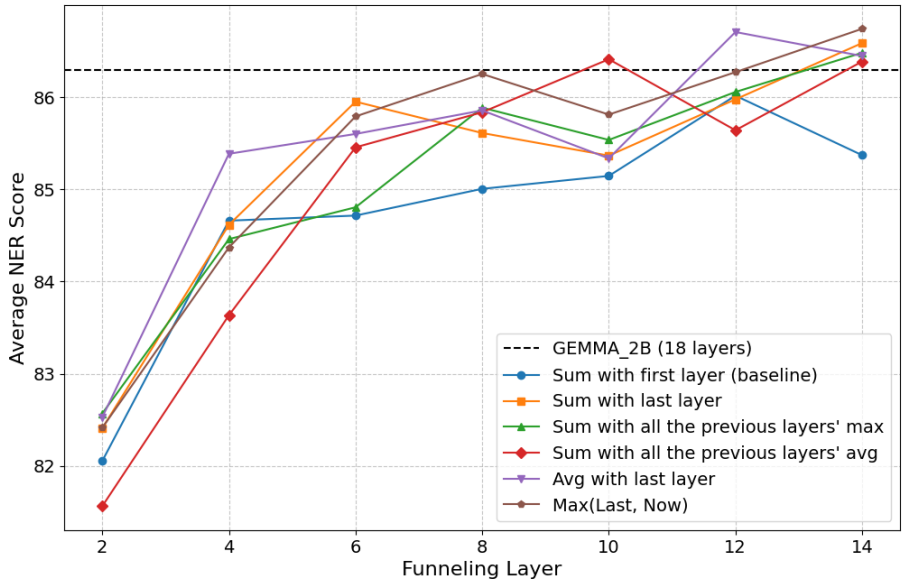

Figure 5: The effect of different recovery operations on NER performance.

uration is further adjusted. This recovery might be indicative of the funnel mechanism's potential to mitigate overfitting in the full models. In fact, the observed improvement in performance at higher funnel recovery layers could be a consequence of reduced overfitting, a phenomenon that has been reported in the literature in contexts such as model quantization (Biderman et al., 2024), where modifying the model architecture can sometimes lead to an increase in performance.

6.2 Information Bottlenecks

Peppered throughout our results are the effect of information bottlenecks: that is, the extend to which information is restricted and its impact on performance.

First, **larger models are impacted more by funneling.** That is, funneling exhibits a notably more pronounced negative effect on the larger and more complex Gemma 7B model compared to Gemma 2B. Due to its increased width (number of neurons per layer) and deeper architecture (10 additional layers), the same funnel operation causes greater information restriction when applied to Gemma 7B. As an example, funneling at the same recovery layer — for instance, layer 6 — aggregates significantly more activations in Gemma 7B due to its wider layers, and the compressed information propagates across more layers (28 versus 18 in Gemma 2B). Consequently, this results in a more substantial performance degradation in Gemma 7B under the same stated funneling operations [3]. This observation indicates that while funneling can offer notable computational speedups by reducing the dimensionality and number of processed activations, careful calibration is required. In particular, the benefits of speedup must be weighed against the degree of performance loss, which can be substantially greater for larger and more complex models.

Additionally, we observed a clear trend that **performance consistently degrades as funneling is introduced at earlier layers**. This phenomenon arises because restricting information early in the network negatively affects the quality of learned representations throughout all subsequent layers. Thus, the timing of the funnel operation significantly influences performance outcomes, underscoring the necessity of strategically selecting later funnel recovery layers if maintaining task accuracy is a priority.

Moreover, among the recovery strategies evaluated, **averaging the unfunneled layer's output with the last layer emerged as the most stable and effective approach**. This averaging operation effectively provides better information passthrough, as it combines the detailed, uncompressed activations from earlier layers with the highly abstract representations in later layers. Compared to other methods such as direct concatenation or

[3] We attribute the superior performance of Gemma 2B to extensive hyperparameter tuning, which wasn't possible for the Gemma 7B model due to limited compute. We however argue that the results are still valid within each model, as both models are compared apples to apples.

max-pooling, averaging ensures a more balanced preservation of both fine-grained details and abstract patterns, which likely explains its superior stability and performance.

7 Limitations

Our study has some limitations worth noting. First, our experiments **exclusively use the dense Gemma model family**, limiting the generalizability of our conclusions to other model types (e.g,, other model families or mixture-of-experts architectures). Additionally, we **do not explore more aggressive funneling configurations**, such as 4-step funneling or multiple funneling layers in one architecture. Finally, we **restrict our study to a single pooling operation,** leaving alternative pooling mechanisms unexamined.

8 Conclusion

Large Language Models (LLMs) present significant computational demands, necessitating ongoing optimization efforts. This study revisit the Funnel Transformer architecture, investigating its application to the contemporary Gemma model family. Experimentation with varied funnel configurations, under both standard and funnel-aware pretraining, on benchmarks like BoolQ, GLUE and WebAnswers, reveals that aggressive funneling creates information bottlenecks, which can degrade performance, particularly in larger models. However, strategic funnel placement and output averaging of compressed and uncompressed layers effectively mitigates these losses. Averaging proves more robust than other recovery methods, likely by balancing detailed and abstract feature integration. Our results underscore the trade-off between computational efficiency and performance in funneling. Future work should explore enhanced funnel-aware pretraining, alternative pooling strategies, and comparisons with other model compression techniques to further optimize LLM efficiency.

8.1 Future Work

To address the identified limitations and deepen our understanding of funneling, several avenues of future research are promising. Expanding the investigation to include diverse model families beyond Gemma could enhance the generalizability of our conclusions. Examining architectures that employ mixture-of-experts could also offer unique perspectives on managing information bottlenecks in complex models. Further exploration into the effects of multiple funnel-aware pretraining layers would provide valuable insights into optimizing funnel configurations. Additionally, exploring alternative pooling operations could identify strategies that mitigate information loss more effectively. Lastly, directly comparing funneling with other established pruning or information bottleneck methods would clarify its relative strengths and weaknesses, guiding more effective deployment in various applications.

8.2 Ethical Considerations

The focus on efficiency should not overshadow broader ethical concerns inherent in LLM development, such as bias amplification, misuse potential, and the environmental impact of training large models, even if inference is optimized. To that end, we argue that more compact and efficient models are a step forward for inference-level environmental concerns and reduce the chance of unintentional detail recitation. We have taken care not to use personally identifiable information in any of our training corpora.

References

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *Preprint*, arXiv:2405.09673. Preprint.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 4271–4282.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun,

Yanqi Zhou, Adams W. Yu, Orhan Firat, et al. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39. Originally presented at arXiv:2101.03961 (2021).

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. *Preprint*, arXiv:2104.08253.

Nadia Mushtaq Gardazi, Ali Daud, Muhammad Kamran Malik, Amal Bukhari, Tariq Alsahfi, and Bader Alshemaimri. 2025. Bert applications in natural language processing: a review. *Artificial Intelligence Review*, 58(6):1–49.

Zixian Huang, Xinwei Huang, Ao Wu, Xiaxia Wang, and Gong Cheng. 2025. Transforming decoder-only models into encoder-only models with improved understanding capabilities. *Knowledge-Based Systems*, 309:112907.

Yizhang Jin, Jian Li, Yexin Liu, Tianjun Gu, Kai Wu, Zhengkai Jiang, Muyang He, Bo Zhao, Xin Tan, Zhenye Gan, et al. 2024. Efficient multimodal large language models: A survey. *Preprint*, arXiv:2405.10739.

Jiedong Lang, Zhehao Guo, and Shuyu Huang. 2024. A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231. IEEE.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*. Association for Computational Linguistics. Also available as arXiv preprint arXiv:2402.14800. Pages to be added upon publication.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *Preprint*, arXiv:cs/0306050. Preprint.

Noam Shazeer, Azalia Mirhoseini, Mariusz Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Xinping Zhao, Baotian Hu, Yan Zhong, Shouzheng Huang, Zihao Zheng, Meng Wang, Haofen Wang, et al. 2024. RaseRec: Retrieval-augmented sequential recommendation.

Xinping Zhao, Yan Zhong, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Dongfang Li, Baotian Hu, and Min Zhang. 2025. FunnelRAG: A coarse-to-fine progressive retrieval paradigm for RAG. *Preprint*, arXiv:2410.10293. Preprint.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Weizhu Chen, and Jianfeng Gao. 2022. Taming sparsely activated transformers with stochastic experts. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*.

A Appendix

Please see Tables 2 and 3 for Gemma2 2B and 7B hyperparameters, respectively.

Table 2: Hyperparameters for Gemma 2B Model Training

Hyperparameter	Value
Gemma Model	GEMMA_2B
Use Pooler	true
Pooling Type	ATTENTION_POOLING
Number of Attention Pooling Heads	4
Number of Transformer Pooling Layers	1
Projection Dimension	None
Max Steps	2000
Max Learning Rate	0.00003
Minimum Learning Rate Fraction	0.1
Warmup Steps	100
Funnel Pooling Config	(1,1,1,1,1,1,2)
Sequence Length	512
Training Batch Size	128
Eval Batch Size	256
Number of Microbatches	0
Weight Decay Rate	None
Soft Labels	False
Scoring BF16 Mode	True
Overtrain Multiplier	1

Table 3: Hyperparameters for Gemma 7B Model Training

Hyperparameter	Value
Gemma Model	GEMMA_7B
Use Pooler	true
Pooling Type	ATTENTION_POOLING
Number of Attention Pooling Heads	4
Number of Transformer Pooling Layers	1
Projection Dimension	None
Max Steps	4000
Max Learning Rate	0.00001
Minimum Learning Rate Fraction	0.1
Warmup Steps	100
Funnel Pooling Config	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2)
Sequence Length	512
Training Batch Size	128
Eval Batch Size	256
Number of Microbatches	0
Weight Decay Rate	None
Soft Labels	False
Scoring BF16 Mode	True
Overtrain Multiplier	1