Language Modeling with Learned Meta-Tokens

Alok N. Shah [*1] **Khush Gupta** [*2] **Keshav Ramji** [*3] **Pratik Chaudhari** [1]

Abstract

While modern Transformer-based language models (LMs) have achieved major success in multi-task generalization, they often struggle to captures long-range dependencies within their context window. This work introduces a novel approach using meta-tokens, special tokens injected during pre-training, along with a dedicated meta-attention mechanism to guide LMs to use these tokens. We pre-train a language model with a modified GPT-2 architecture equipped with meta-attention over less than 100B tokens, achieving strong performance on a suite of synthetic tasks. We suggest that these gains arise due to the meta-tokens *sharpening* the positional encoding, operating as content-based landmarks, implicitly compressing preceding context and "caching" it in the meta-token. At inference-time, the meta-token points to relevant context, facilitating length generalization. Our findings suggest that pre-training LMs with meta-tokens offers a simple, data-efficient method to enhance long-context language modeling performance, while introducing new insights into their behavior towards length generalization.

1. Introduction

Transformer-based language models (LMs) have showcased remarkable capabilities across diverse language tasks (Brown et al., 2020b; Chowdhery et al., 2022; OpenAI, 2023). Nevertheless, such models suffer from an inability to capture dependencies spanning over their entire context window. With growing adoption and ever-expanding demands on the context over which the model can process and reason, it is vital to develop methods that facilitate long-context adaptation and length generalization. In this work, we propose a simple solution, by way of *meta-tokens*, learned tokens periodically injected into the input sequence during pretraining, and cleverly placed during fine-tuning. Unlike conventional dummy tokens (Goyal et al., 2024), meta-tokens are explicitly trained via a dedicated sparse attention layer, guiding the model to condense and "cache" contextual information as an in-line storage mechanism. As a result, these tokens act as adaptive landmarks (Mohtashami & Jaggi, 2023), summarizing preceding context segments into compact representations. At inference time, meta-tokens provide implicit pathways to distant information, enabling models to generalize effectively across sequences longer than those encountered during training.

We demonstrate the empirical efficacy of this approach by pre-training a 152M parameter modified GPT-2 model with meta-tokens. Specifically, we show that our method drastically improves performance on synthetic tasks explicitly designed to test fundamental abilities over tasks such as recall and copying. We trace these gains to a subtle mechanism: meta-tokens provably induce a *sharpening* effect on positional encoding, enabling the meta-token to locate its position based on the content it stores and reducing the entropy of the attention distribution. We present evidence that this sharpening is responsible for an anchoring effect on relevant distant tokens, facilitating robust length generalization.

2. Training Language Models with Meta-Attention

We introduce a set of M meta-tokens (denoted as m); given a context length or block size of the model, n, we take $M = kn$ for some constant fraction $k \in [0, 1]$[1]. The aim of introducing these meta-tokens is to capture or store contextual information to enhance the model's retrieval and reasoning capabilities; attending to a meta-token should enable implicit retrieval of the context that it stores, guiding shortcut paths over the context window. In practice, these may be treated akin to adding a filler token to the model's vocabulary.

The M tokens are injected into the input sequences dur-

[*]Equal contribution [1]Department of Electrical and Systems Engineering, University of Pennsylvania [2]Department of Computer and Information Science, University of Pennsylvania [3]IBM Research AI. Correspondence to: Alok N. Shah <alokshah@upenn.edu>, Khush Gupta <khushg@upenn.edu>.

Proceedings of the 2^{nd} Workshop on Long-Context Foundation Models, Vancouver, Canada. 2025.

[1]We take $k = 0.1$ in practice; balancing next-token prediction over the standard vocabulary while injecting a non-trivial number of meta-tokens.

ing pre-training uniformly at random, which was informed by two key premises. While we desire interpretability and control in applying these tokens, and as a result, prefer distinguishability at the task level, this is challenging to do without explicitly fixing a downstream task, impeding generality. The second consideration was in how they specifically they should be injected. While (Zelikman et al., 2024) introduced $<|\text{startofthought}|>$ and $<|\text{endofthought}|>$ tokens interleaved between reasoning steps near punctuation (serving as natural break), the introduction of a rough periodicity between tokens during pre-training could result in being trapped into local minima in the optimization landscape. We instead chose to follow the random injection scheme, supported by the meta-token pre-training approach outlined in (Goyal et al., 2024).

We ensure that the trained model incurs no loss for predicting meta-tokens, unlike a standard token in the vocabulary – the meta-tokens' indices are simply shifted and removed when computing the binary cross-entropy (BCE) loss.

Meta-Attention Mechanism. We augment our transformer H to take P which contains the positions of the meta-tokens. We introduce a sparse attention mechanism, called meta-attention, which selectively modifies attention scores for the specially marked "meta-tokens" within a sequence. This allows the model to simulate selective attention, influencing the final behavior by focusing on these meta-tokens.

Let the indices of special "meta-tokens" be denoted by positions $\in \mathbb{R}^{B\times T'}$, where T' is the number of meta tokens in a batch. We construct a meta mask $P \in \mathbb{R}^{B\times T\times T}$ to influence the attention mechanism. For each batch element b and token positions i, j:

$$P[b,i,j] = \begin{cases} 0 & \text{if both } i \text{ and } j \text{ are meta tokens} \\ -\infty & \text{otherwise} \end{cases}$$

The meta-attention operation is defined as:

$$\text{MetaAttn}(Q,K,V) = \text{softmax}\left(\left(\frac{QK^\top}{\sqrt{d_k}} + M\right) + P\right)V$$

Where M is the same causal mask as before. Here, the meta mask P allows attention to flow only among the meta tokens in the sequence, introducing a distinct interaction compared to regular attention. This meta-attention layer selectively modifies the attention by influencing the flow of information to and from these meta tokens, distinguishing itself from the standard causal attention.

To assemble the architecture used for our model, we insert the meta-attention mechanism after the causal masked self-attention computation, to specifically attend to the injected meta tokens, as defined above. We provide a complete breakdown of the architecture in Appendix B.

3. Meta-Tokens Sharpen Positional Encoding

Model Training and Architecture. All experiments were performed with 4 NVIDIA A100 GPUs, training the meta attention transformer for 200,000 iterations or 98B tokens using Distributed Data Parallel (DDP) on the Colossal Cleaned Crawl Corpus (C4) (Raffel et al., 2020). The configuration and hyperparameters used in our pre-training are included in Appendix B and C. As a baseline, we also pre-train GPT-2 (124M) on C4, with identical hyperparameters. The primary change from a standard GPT-2 architecture is the addition of RoPE to enable better generalization to longer contexts and improve stability in next-token prediction tasks.

Long Context Extension We extend our transformer model's context window from 1024 tokens to longer sequences by training two distinct models with context lengths of 4096 and 8192 tokens, respectively. This extension is implemented using the YaRN method (Peng et al., 2023), which dynamically scales Rotary Positional Embeddings (RoPE) to effectively process significantly longer sequences without compromising performance or computational efficiency. The key parameters are detailed in Appendix D

Experimental Setup and Tasks. We design four synthetic tasks to evaluate the recall capabilities of models trained with meta-tokens. The tasks are *List Recall*, *Segment Counting*, *Parity*, and *Copying*. For each task, we define three difficulty levels by varying the maximum sequence length. In all tasks, we insert a designated `_PAUSE_` meta-token at task-specific positions to indicate where the model should focus its meta-attention. We fine-tune on synthetic data that we generate for each task (binned by instance length) and report the validation score on a held-out test set. Detailed examples for each task are provided in Appendix K.

- **List Recall:** Given N named lists of length k, the model is prompted to recall a specific item from a specified list. We insert a `_PAUSE_` meta-token immediately following the list containing the queried item, as well as before the final question. The expected answer is the corresponding item. Task difficulty is scaled by varying the list length k and number of lists N.
- **Segment Counting:** The model is presented with several named lists, with a segment in these lists wrapped by by `_PAUSE_` meta-tokens. The prompt then asks how many times a specified item appears between the two meta-tokens. The task difficulty changes based on the number and size of the named lists.

The descriptions for parity and copying are in Appendix F.2.3 and F.2.4 respectively. Within these tasks, we inves-

tigate length generalization by fine-tuning our model in multiple phases. At each phase, we assess the model's performance on sequence lengths exceeding those seen during that phase's training, enabling us to evaluate its generalization to longer contexts. In addition, Appendix F reports the performance of our models on a context length of 2048 tokens, which is twice the length seen during pretraining (1024 tokens).

Baselines. For a controlled comparison, we also pre-train a GPT-2 model (NanoGPT, 124M; (Karpathy, 2023)) on C4, with identical hyperparameters as the meta-tokens model. Additionally, we use Eleuther AI's GPT-Neo-125M (Black et al., 2021) as another baseline.

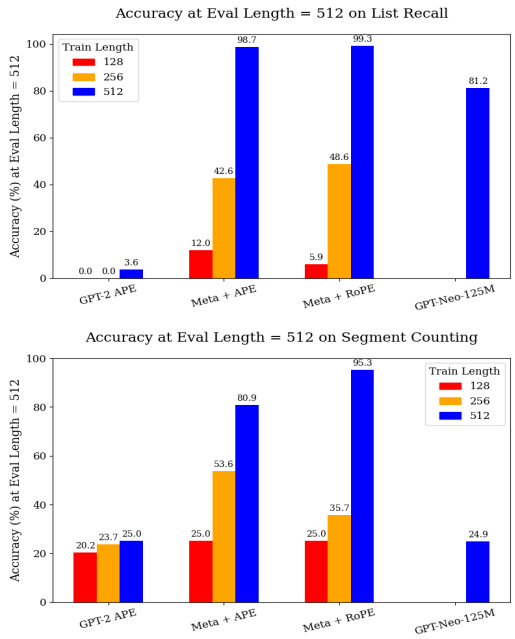


Figure 1. We study the performance of the pre-trained GPT-2 w/ APE, and Meta-attention {w/ APE, w/RoPE}, all fine-tuned on synthetic data for their respective tasks at the maximum train lengths indicated in the legends. All experiments are performed on a test set of prompt lengths up to 512 tokens.

Meta-Tokens Improve Recall and Length Generalization. As shown in Figure 1, models utilizing meta-tokens consistently outperform baseline GPT-2 and GPT-Neo-125M models across both tasks and training lengths. Notably, GPT-2 trained with absolute positional embeddings (APE) generally performs poorly except in segment counting and parity tasks, suggesting potential improvements with additional training data and highlighting the data efficiency of meta-tokens models. Importantly, our meta-token models improve more rapidly with increased training length compared to GPT-2 models and significantly surpass GPT-Neo-125M performance despite GPT-Neo being pre-trained on nearly triple the data volume. We further explore the effect of positional information by ablating positional encoding and text embeddings specifically at meta-token indices (Tables 10-13). Surprisingly, removing positional encoding alone generally matches or improves model accuracy compared to the original setup, with the segment counting task being the notable exception. Conversely, eliminating token embeddings substantially decreases performance on tasks like List Recall, Segment Counting, and Copying, indicating their critical role. Specifically, performance on Segment Counting tasks significantly increases at longer training lengths, e.g., by $+28.6\%$ with APE and $+10.7\%$ with rotary positional embeddings (RoPE), compared to only $+3.5\%$ improvement in GPT-2. On extended test lengths (up to 1024 tokens), zeroing out positional encodings at meta-token indices further improves generalization, notably boosting List Recall task performance by up to $+38.9\%$. Table 2 exhibits a similar trend for the YaRN models, achieving strong performance across its respective context windows, and even achieves non-trivial accuracy beyond the window. Fine-tuning the 8k YaRN model on examples of up to a length of 4k can generalize very well up to 8k. These findings underscore the substantial advantages of training with meta-tokens and the nuanced role positional encoding plays in task-specific and length-generalization contexts.

Model (Split, Train Len)	Full	No Pos	Δ(pp)
Meta + APE (medium, 128)	77.8%	88.9%	+11.1
Meta + APE (hard, 128)	11.1%	22.2%	+11.1
Meta + APE (extra-hard, 512)	11.1%	50.0%	+38.9
Meta + RoPE (medium, 128)	44.4%	55.6%	+11.1
Meta + RoPE (hard, 256)	33.3%	66.7%	+33.3
Meta + RoPE (extra-hard, 256)	0.0%	22.2%	+22.2
Meta + RoPE (extra-hard, 512)	44.4%	55.6%	+11.1

Table 1. Configurations where zeroing the positional encoding at inference improves List Pointer accuracy (Δ(pp): % points)

3.1. Examination of the PE Sharpening Effect

As discussed above, the results in Tables 10-13 suggest that the positional encoding of the meta-token can potentially be holding back the downstream performance of the meta-attention models. We posit that the model is instead relying on its content – cached context stored within the meta-token – to *sharpen* its sense of its position in the sequence.

Next, we aim to formally define this notion of sharpness in the context of positional encoding, and its relationship to the model's logits. Let $\alpha_{i\to k} = \text{softmax}_k(Q_i K_j^T + b_{i-j})$ be the attention distribution for query i over keys j, with relative bias term b_{i-j}. We define the *sharpness* of the positional encoding by the entropy: $H(\alpha_i) = -\sum_j \alpha_{i\to j} \log \alpha_{i\to j}$.

Intuitively, when a meta-token is present at position t, the

Table 2. Token Accuracy (%) on List Recall and Count Segment tasks across evaluation context lengths.

Task	(Train, Finetune)	2k	3k	4k	5k	6k	7k	8k	10k	12k	14k	16k
List Recall	(4k, 2k)	19.5	16.0	13.7	0.9	0.0	0.0	0.9	1.1	0.0	2.1	1.1
	(4k, 4k)	85.0	88.2	90.2	20.5	1.8	1.0	3.5	4.4	1.1	2.1	2.1
	(8k, 4k)	85.0	95.8	91.2	97.4	98.2	96.2	93.9	31.9	0.0	2.1	2.1
	(8k, 8k)	92.9	98.3	97.1	100.0	98.2	100.0	100.0	89.0	26.1	10.4	9.6
Count Segment	(4k, 2k)	19.1	23.8	19.2	14.6	25.2	14.1	14.0	12.0	16.0	8.0	6.0
	(4k, 4k)	17.5	23.8	31.8	20.3	30.4	19.3	19.1	14.0	26.0	12.0	16.0
	(8k, 4k)	19.1	23.8	14.3	11.1	20.6	12.7	12.7	14.0	16.0	14.0	12.0
	(8k, 8k)	27.0	33.3	15.9	19.1	27.0	19.1	23.8	22.0	18.0	18.0	18.0

model's attention becomes peaked around a small set of keys; this "honing in" behavior reduces $H(\alpha)$ compared to APE or RoPE without meta-tokens. In this manner, meta-tokens behave as **content-driven landmarks**, serving as a low-entropy channel that points to relevant context.

The full proof of Theorem 3.1 is included in Appendix J.

Theorem 3.1. *Consider a Transformer head at query position i over keys $1, \ldots, N$. Let $\alpha_i^{abs}(j) \propto \exp(Q_i K_j^T)$ be the attention under absolute positional encoding and let $\alpha_i^{meta} \propto \exp(Q_i K_j^T + \delta_{j,j^*}\Delta)$ when a meta-token at position j^* introduces an additive logit boost of $\Delta > 0$. Then, for some function $\kappa(\Delta) > 0$, $H(\alpha_i^{meta}) \leq H(\alpha_i^{abs}) - \kappa(\Delta)$.*

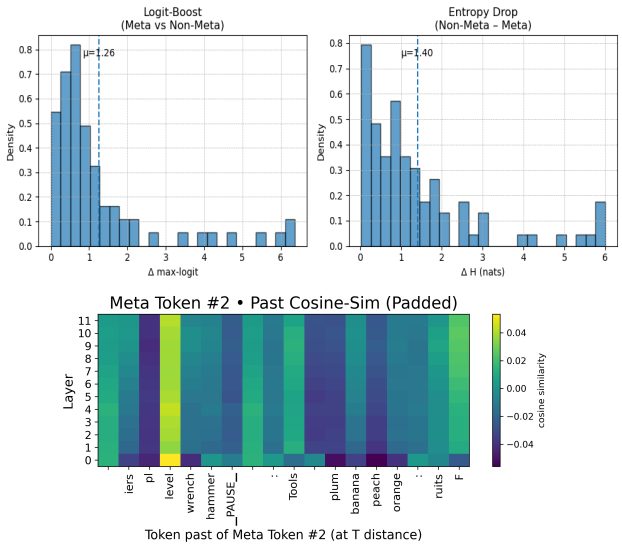


Figure 2. (Top) We analyze the change in logits at the meta-token position after zeroing TE (left) and show that boosted logits correspond with reduced entropy over the softmax of the logits (right). (Bottom) We study the cosine similarity over the token embeddings, and observe spikes, diminishing as we move further away, confirming our claims of implicit compression via "caching".

We note that this theorem also applies to RoPE, using $\alpha_i^{\text{RoPE}}(j) \propto \exp Q_i(\text{RoPE}(K_j))^T$. A natural consequence of Theorem 3.1 is that the meta-token operates as an "anchor" over the logits by creating a margin Δ that concentrates the softmax. Thus, any learned meta-token embedding – provided that it boosts the logits at j^* – guarantees sharper attention by reducing that attention head's entropy.

In Figure 2, we analyze the logits, comparing two settings: (1.) the current meta-token and (2.) the meta-token with its token embedding zeroed out. We find that the former gains a sizable amount over the latter, reinforcing the additive logit boost assumption made in Theorem 4.1. Empirically, we show that the entropy over the softmax distribution of the logits decreases, thus corroborating Theorem 3.1.

4. Discussion and Conclusion

Our findings suggest that decoder-only language models trained with meta-tokens and meta-attention achieve strong performance on recall tasks. Furthermore, they exhibit length generalization, with performance improvements when ablating positional encoding at the meta-tokens. We suggest that hybrid attention methods such as RNoPE (Yang et al., 2025) could be suitable for facilitating long-context modeling with meta-tokens. With meta-tokens operating like anchors within the context, it would be valuable to explore the impact of our proposed mechanism in pre-training larger models over longer context windows, under greater computational resources. Our synthetic tasks are designed to test length generalization for recall, as an indication of long-context modeling capabilities; training larger models would validate its potential for real-world deployment.

We introduce *meta-tokens* in language model pre-training, with dedicated meta-attention mechanism which learns the relationship between standard and meta-tokens. This improves performance and length generalization on synthetic recall tasks, even without positional encoding. We provide evidence to suggest that the meta-tokens sharpen the positional encoding, operating as contextual landmarks by implicitly compressing preceding context. These phenomena demonstrate the promise of long-context language modeling enabled via data-efficient pre-training using meta-tokens.

5. Acknowledgments

The authors would like to thank Surbhi Goel for valuable discussions which have formed the basis of our study and support with computational resources. We would also like to thank Ben Keigwin for helpful conversations and suggestions towards designing our analysis. This work was supported in part by a research compute grant from Lambda Labs.

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyxQzBceg.

Black, S., Leo, G., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715. If you use this software, please cite it using these metadata.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Brown, T. B., Mann, B., Ryder, N., et al. Language models are few-shot learners. *NeurIPS*, 2020b.

Burtsev, M. S., Kuratov, Y., Peganov, A., and Sapunov, G. V. Memory transformer, 2021. URL https://arxiv.org/abs/2006.11527.

Chowdhery, A., Narang, S., Devlin, J., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954.

Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., and Nagarajan, V. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

Grattafiori, A. et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Haviv, A., Ram, O., Press, O., Izsak, P., and Levy, O. Transformer language models without positional encodings still learn positional information. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1382–1390, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.99. URL https://aclanthology.org/2022.findings-emnlp.99/.

Jiang, W., Zhang, J., Wang, D., Zhang, Q., Wang, Z., and Du, B. Lemevit: Efficient vision transformer with learnable meta tokens for remote sensing image interpretation. In Larson, K. (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 929–937. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/103. URL https://doi.org/10.24963/ijcai.2024/103. Main Track.

Karpathy, A. nanoGPT. https://github.com/karpathy/nanoGPT, 2023. Accessed: 2025-05-16.

Kazemnejad, A., Padhi, I., Natesan, K., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Drrl2gcjzl.

Marshall, A. W., Olkin, I., and Arnold, B. C. *Inequalities: Theory of Majorization and Its Applications*. Springer Series in Statistics. Springer New York, New York, NY, 2 edition, 2011. ISBN 978-0-387-68276-1. doi: 10.1007/978-0-387-68276-1.

Mohtashami, A. and Jaggi, M. Random-access infinite context length for transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=7eHn64wOVy.

OpenAI. Gpt-4 technical report. https://openai.com/research/gpt-4, 2023.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models, 2023. URL https://arxiv.org/abs/2309.00071.

Pfau, J., Merrill, W., and Bowman, S. R. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation, 2022. URL https://arxiv.org/abs/2108.12409.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. OpenAI Technical Report, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Accessed: 2025-05-15.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In Walker, M., Ji, H., and Stent, A. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074/.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

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

Yang, B., Venkitesh, B., Talupuru, D., Lin, H., Cairuz, D., Blunsom, P., and Locatelli, A. Rope to nope and back again: A new hybrid attention strategy, 2025. URL https://arxiv.org/abs/2501.18795.

Zelikman, E., Harik, G. R., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. Quiet-STar: Language models can teach themselves to think before speaking. In *First*

Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=oRXPiSOGH9.

A. Preliminaries

Causal Multi-Head Attention. Let $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ denote an input sequence of tokens of length T, $\mathcal{V}$ denote the vocabulary size of V, and $E : \mathcal{V} \to \mathbb{R}^d$ represent the the token embedding function mapping each token to a d-dimensional vector. Each x_t is embedded into some continuous representation where $\mathbf{e}_t = E(x_t) + \mathbf{p}_t$, such that $\mathbf{p}_t$ is the positional encoding for t.

In decoder-only architecture, we utilize causal self-attention to ensure that predictions for a given token are only based on preceding tokens. The causal self-attention mechanism modifies the attention computation by masking future positions in the attention weights. Formally:

$$\text{Causal Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)$$

where M masks future tokens, ensuring that the model can only attend to current and past tokens. If A is the matrix of attentions scores, then

$$A_{ij} = \begin{cases} \text{softmax}(A_{ij}) & \text{if } i \geq j \\ 0 & \text{if } i < j \end{cases}$$

This masking zeros attention scores for future tokens, allowing only the relevant past tokens to influence the current token's representation.

Positional Encoding. Positional encoding was introduced in Transformer pre-training to provide models with information about the ordering of tokens. With absolute positional embeddings (APE; (Vaswani et al., 2017)), each position t in the sequence receives a vector p_t, independent of its content, so tokens are distinguished in an index-by-index manner. Given learned token-embedding lookup table $E : V \to \mathbb{R}^d$ for vocabulary V and hidden dimension d, and positional embedding $p_t = \text{Emb}_{pos}(t)$ for $t \in [0, T-1]$ and $\text{Emb}_{pos} \in \mathbb{R}^{T \times d}$. Each token embedding is then defined as $e_t = E(x_t) + p_t$; this method was used in GPT-2 and GPT-3 (Radford et al., 2019; Brown et al., 2020a).

By contrast, Rotary Position Embedding (RoPE; (Su et al., 2023)) rotates each pair of embedding dimensions by an angle proportional to position, rather than adding a separate vector per position. This makes the difference in attention scores directly encode relative distance between embeddings. The hidden vector h is split into $\frac{d}{2}$ contiguous 2-D slices, and the angle for a position t is defined as $\theta_{t,i} = \frac{t}{10000^{2i/d}}$. The 2-D rotation matrix is taken as $R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. Then, $\text{RoPE}(h)_t^{(2i:2i+1)} = R(\theta_{t,i})h^{(2i:2i+1)}$. This has proven successful in the Llama models (Grattafiori et al., 2024). It can be observed that RoPE reflects the relative offset $i - j$, with the dot product $\langle Q_i, K_j \rangle$ introducing a new factor of $\cos\left(\frac{i-j}{10000^{2i/d}}\right)$. This is reflected in works using *relative bias* (Shaw et al., 2018), which introduces a bias term as a learned function over the $i - j$ distance. T5 (Raffel et al., 2020) then adds this bias to $\langle Q_i, K_j \rangle$.

B. Full Architecture Details

We provide a full outline of the architecture design out method uses. Our architecture is equivalent to the NanoGPT (GPT-2) architecture, while introducing the meta-attention block after the initial causal masked attention and layer normalization computation.

1. **Input Layer:** Given an input sequence of tokens $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$, we first embed each token into a continuous representation. Instead of absolute positional encodings, we apply Rotary Position Embeddings (RoPE) (Su et al., 2023) to inject positional information. For each token, the embedded representation is:

   $$\mathbf{e}_t = \text{RoPE}(E(x_t), t),$$

   where $\text{RoPE}(\cdot, t)$ denotes the rotary positional embedding applied to the t^{th} position, with a base $\theta = 10000.0$.

2. **Causal Masked Self-Attention:** The first layer consists of the causal masked self-attention mechanism. For each head h, the attention operation is computed as:

   $$\text{CausalAttention}_h(Q, K, V) = \text{softmax}\left(\frac{QK_h^\top}{\sqrt{d_k}} + M\right)V_h,$$

where Q, K, V are the query, key, and value matrices derived from the input embeddings $\mathbf{E}$, and M is the mask matrix.

3. **Meta Attention Layer:** After the causal masked self-attention, we integrate the meta-attention mechanism to specifically attend to the injected meta tokens. This operation is defined as:

$$\text{MetaAttention}(Q, K, V, P) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M_{\text{causal}} + P\right)V,$$

where P is the meta mask constructed from the indices of the meta tokens.

4. **Feedforward Layer:** Following the attention layers, we pass the output through a feedforward neural network defined by:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2,$$

where W_1, W_2 are weight matrices, and b_1, b_2 are bias vectors.

5. **Layer Normalization:** After both the causal self-attention and meta-attention operations, we apply layer normalization:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma + \epsilon},$$

where μ and σ are the mean and standard deviation of the features, and ϵ is a small constant for numerical stability.

6. **Final Output Layer:** The final layer projects the output of the last feedforward layer back to the vocabulary size to produce the s for the next token prediction:

$$\text{s} = \text{softmax}(xW_{\text{out}} + b_{\text{out}}),$$

where W_{out} and b_{out} are the output weight matrix and bias vector, respectively.

C. Pre-training Hyperparameters and Model Details

Our decoder-only modified GPT-2 model was pre-trained on the C4 dataset with the following configuration and hyperparameters:

Table 3. Pretraining Configuration Parameters

Parameter	**Value**
Batch Size	12
Gradient Accumulation Steps	40
Block Size	1024
Number of Layers	12
Number of Heads	12
Embedding Size	768
Learning Rate	6e-4
Weight Decay	1e-1
Max Iterations	600,000
Warmup Iterations	2,000
Minimum Learning Rate	6e-5
Dropout Rate	0.0
RoPE Theta	10000.0
Initial Model	Resume
Optimizer	AdamW
AdamW Beta1	0.90
AdamW Beta2	0.95
Gradient Clipping	1.0
Tokenizer	tiktoken

D. YaRN Hyperparameters

Parameter	4096-token model	8192-token model
yarn_scale	4.0	8.0
yarn_original_max_seq_len	1024	
yarn_extrapolation_factor	1.0	
yarn_attn_factor	1.0	
yarn_beta_fast	32.0	
yarn_beta_slow	1.0	

Table 4. YaRN parameter configurations for extended context models.

E. Related Work

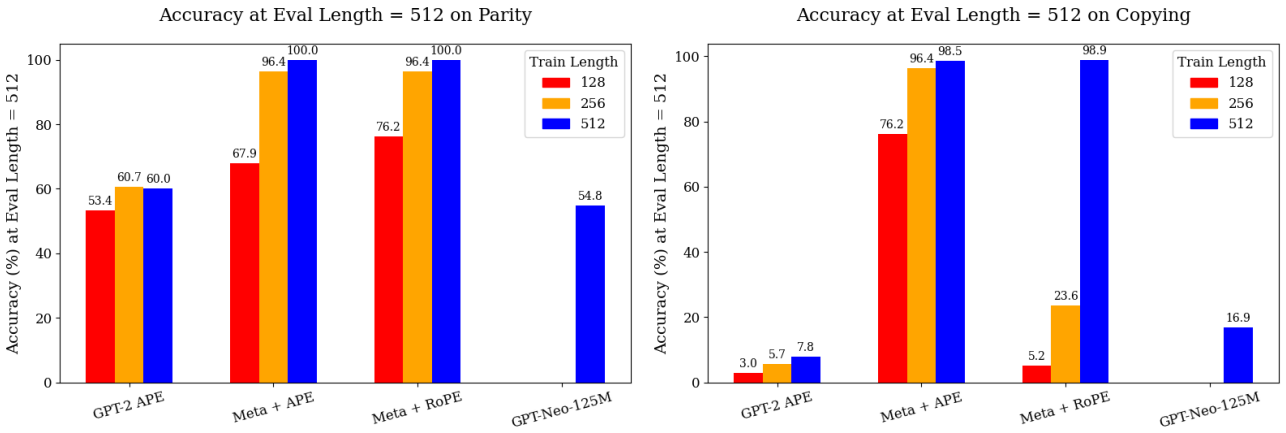
Pause and Memory Tokens As detailed in our work, recent studies on Transformer-based models have explored the introduction of special tokens, beyond ordinary vocabulary symbols. *Pause* or *dummy* tokens as introduced in (Goyal et al., 2024) enhance computational width, allowing models to perform additional internal computation by effectively delaying their outputs. This yields empirical gains on question answering and reasoning-intensive tasks. Similarly, (Pfau et al., 2024) explore using filler tokens – sequences of seemingly meaningless symbols – as a stand-in for chain-of-thought. These special tokens may also delineate phases of reasoning, as in Quiet-STaR (Zelikman et al., 2024). Quiet-STaR uses a begin-of-thought token and an end-of-thought token, generating a silent rationale sequence for each step before emitting the next word, showing that this helps zero-shot reasoning.

Works such as Memory Transformer (Burtsev et al., 2021) and Landmark Attention (Mohtashami & Jaggi, 2023) introduce memory tokens; the former prepends them, while the latter uses them as learnable keys for retrieval over blocks of context. Our work is most closely related to the latter, while performing this retrieval in a purely implicit manner via the observed ”pointer” mechanism. For vision transformers (ViTs), LeMeVit (Jiang et al., 2024) introduces a similar meta-tokens notion as our work by adding learnable sparse tokens and an attention mechanism between standard tokens and their meta tokens, improving performance and reducing spatial redundancy. (Darcet et al., 2024) uses specialized ”register” tokens applies to patches to denoise images by extracting the high-norm, outlier tokens, smoothening the feature and attention maps. These works suggest that special tokens, even devoid of semantic content, can influence a model's internal reasoning and memory mechanisms.

Positional Encoding We have already described absolute positional embeddings (APE), rotary positional embeddings (RoPE) and relative bias in Section A. In addition to these methods, ALiBi (Press et al., 2022) adds a fixed linear penalty to attention scores based on the distance between query and key positions, favoring nearer tokens and generalizing to longer contexts with minimal loss in perplexity. Recent work has suggested that Transformers without any added position embeddings can still learn order information and, in some cases, generalize to longer sequences better than models with standard positional encoding. NoPE (Kazemnejad et al., 2023) showed that models trained without positional embeddings can achieve strong length extrapolation in comparison to models trained with positional encoding. They can internally represent both absolute and relative PEs without any explicit positional signal, suggesting these may emerge implicitly via training dynamics or over the data distribution. NoPos (Haviv et al., 2022) also found a similar result, suggesting that models trained without PE can infer their absolute position due to causal attention masks. These findings are highly relevant to our work, given our evidence on length generalization behavior whiling zeroing the positional encoding at the meta-tokens.

F. Additional Experimental Details

F.1. Figures on Parity and Copying



F.2. Synthetic Data Generation

We generate 90,000 train examples and held-out test set of 10,000 examples for each task.

F.2.1. List Recall

We generate a suite of "list-pointer" examples by sampling random categories and list items, inserting a special meta token as a marker, and asking the model to recover the item immediately following the meta-token. Each example consists of:

1. m categories drawn without replacement from a fixed set of 20.
2. n items per category, sampled with replacement from the category's 10–item inventory
3. One "target" category in which we inject a single meta token after the jth item ($j \in [n]$) and then append the remaining items
4. A question line "Q: What is item j of ¡target¿? _META_"

This pipeline yields curriculum-structured data that systematically probes the model's ability to attend to and copy items in long, multi-list contexts.

Phase	**m** (Num. categories)	**n** (List length)	**Approx. prompt-token range**
1	Uniform 3–8	Uniform 3–10	Short ($\approx$ 100–200 tokens)
2	Uniform 8–12	Uniform 3–16 (bimodal)	Mid-range ($\approx$ 200–300 tokens)
3	Uniform 12–19	Mixture $\{3$–$8, 9$–$16, 17$–$25\}$	Full-range ($\approx$ 500–700 tokens)
4	Uniform 15–20	Uniform 40–60	"Extra-hard" $\leq$ 1024 tokens
5	Uniform 15–20	Uniform 90–110	"Long" $\leq$ 2048 tokens

Table 5. Curriculum schedule for synthetic data.

F.2.2. Segment Counting

Similar to List-pointer, except the model must count occurrences of a target token within a meta-token bracketed list segment. Uses the schedule dictated by Table F.2.1. Asks the question: "Q: How many times does ¡token¿ appear between the pauses around ¡Category¿? _META_".

F.2.3. PARITY

Generates examples where the model computes the XOR (parity) of a bit-string segment up to the first L characters where L is drawn phase-dependently. the same scheduling dictated by Table F.2.1. Asks the question: "Q: What is the XOR of all bits before this pause? _META_ "

F.2.4. COPYING

Generates examples where the model must copy a bracketed span from a text. Uses schedule dictated by Table F.2.1 and samples an additional copy length C and distance length D depending on the phase

G. Licenses

nanoGPT

Our implementation of the vanilla GPT-2 is based on the nanoGPT repository (`https://github.com/karpathy/nanoGPT`), which is licensed under the MIT License.

EleutherAI GPT-Neo-125M

We directly use the EleutherAI GPT-Neo 125M model checkpoint and weights, available via the Hugging Face Model Hub at `https://huggingface.co/EleutherAI/gpt-neo-125m`. This model is released under the MIT License.

C4 Dataset

Our model was trained on the C4 dataset (`https://huggingface.co/datasets/allenai/c4`), which is provided under the Open Data Commons Attribution License (ODC-BY).

tiktoken

We use the `tiktoken` library from OpenAI for tokenization (`https://github.com/openai/tiktoken`), which is released under the MIT License.

H. Complete Experimental Results

H.1. Synthetic Task Accuracies Across Test Lengths

We stress test RoPE models at a sequence length of 2048—twice the pretraining block size of 1024—as relative position embeddings naturally support extrapolation beyond the training context window. In contrast, absolute positional encodings (APE) cannot generalize to sequences longer than those seen during pretraining.

Table 6. Accuracy (%) across evaluation lengths for each model train on List Recall

Model (Train Length)	128	256	512	1024	2048
GPT-2 APE (128)	4.2	1.2	0.0	0.0	—
GPT-2 APE (256)	6.8	2.4	0.0	0.0	—
GPT-2 APE (512)	19.8	9.5	3.6	0.0	—
Meta + APE (128)	100.0	86.4	12.0	4.1	—
Meta + APE (256)	100.0	98.6	42.6	3.9	—
Meta + APE (512)	100.0	100.0	98.7	11.1	—
Meta + RoPE (128)	100.0	60.7	5.9	0.0	0.0
Meta + RoPE (256)	100.0	100.0	48.6	23.5	0.0
Meta + RoPE (512)	100.0	100.0	99.3	58.9	5.6
GPT-Neo-125M	85.6	86.0	81.2	—	—

Table 7. Accuracy (%) across evaluation lengths for each model trained on Segment Counting. Each model is evaluated on longer contexts than seen during training.

Model (Train Length)	128	256	512	1024	2048
GPT-2 APE (128)	32.1	27.4	20.2	0.0	—
GPT-2 APE (256)	40.3	56.2	23.7	0.0	—
GPT-2 APE (512)	30.1	32.1	25.0	0.0	—
Meta + APE (128)	77.4	55.9	25.0	11.1	—
Meta + APE (256)	83.3	77.4	53.6	22.4	—
Meta + APE (512)	91.7	79.8	80.9	33.3	—
Meta + RoPE (128)	77.4	64.3	25.0	22.7	0.0
Meta + RoPE (256)	64.3	64.3	35.7	33.3	0.0
Meta + RoPE (512)	90.9	91.4	95.3	66.7	11.1
GPT-Neo-125M	31.4	25.9	24.9	—	—

Table 8. Accuracy (%) across evaluation lengths for each model train on Parity

Model (Train Length)	128	256	512	1024	2048
GPT-2 APE (128)	75.0	56.0	53.4	45.2	—
GPT-2 APE (256)	75.0	67.0	60.7	46.2	—
GPT-2 APE (512)	75.0	54.8	60.0	40.5	—
Meta + APE (128)	100.0	75.0	67.9	52.4	—
Meta + APE (256)	100.0	97.6	96.4	69.1	—
Meta + APE (512)	100.0	100.0	100.0	86.7	—
Meta + RoPE (128)	100.0	66.7	76.2	59.5	44.1
Meta + RoPE (256)	97.6	100.0	96.4	61.9	52.4
Meta + RoPE (512)	100.0	100.0	100.0	69.1	63.1
GPT-Neo-125M	80.4	59.1	54.8	—	—

H.2. Ablations on Positional Encoding and Token Embedding

Table 10. Accuracy (%) on the List-Recall task under different ablations: zeroing the positional encoding (No Pos), zeroing the text embeddings (No Embed), or zeroing both of the meta-tokens.

Model (PE)	Full	No Pos	No Embed	Neither
Meta + APE (128)	100.0	99.3	17.4	59.7
Meta + RoPE (128)	100.0	100.0	32.4	24.0
Meta + APE (256)	86.4	86.9	12.2	16.2
Meta + RoPE (256)	100.0	100.0	4.0	6.6
Meta + APE (512)	100.0	100.0	52.1	84.3
Meta + RoPE (512)	100.0	100.0	59.6	25.2

Table 9. Accuracy (%) across evaluation lengths for each model trained on Copying

Model (Train Length)	128	256	512	1024	2048
GPT-2 APE (128)	6.0	5.3	3.0	0.0	—
GPT-2 APE (256)	6.8	6.0	5.7	0.0	—
GPT-2 APE (512)	3.8	4.8	7.8	0.0	—
Meta + APE (128)	100.0	66.7	76.2	2.6	—
Meta + APE (256)	100.0	100.0	96.4	7.9	—
Meta + APE (512)	100.0	100.0	98.5	87.4	—
Meta + RoPE (128)	96.6	73.0	5.2	0.0	0.0
Meta + RoPE (256)	98.2	100.0	23.6	9.3	3.2
Meta + RoPE (512)	99.0	98.9	98.9	89.4	11.8
GPT-Neo-125M	31.5	22.7	16.9	—	—

Table 11. Accuracy (%) on the Segment Counting task under different ablations: zeroing the positional encoding (No Pos), text embeddings (No Embed), or both, only on the meta-token.

Model (Train Length)	Full	No Pos	No Embed	Neither
Meta + APE (128)	77.4	63.1	31.0	47.6
Meta + APE (256)	83.3	88.1	32.1	40.5
Meta + APE (512)	91.7	82.1	34.5	51.2
Meta + RoPE (128)	77.4	70.2	59.5	36.9
Meta + RoPE (256)	64.3	53.6	30.9	30.9
Meta + RoPE (512)	80.9	72.6	36.9	25.0

H.3. Positional Encoding Robustness Ablations

Table 14. Accuracy (%) on the List Pointer task with Gaussian noise added to positional encoding.

Model (Train Length)	Noise 0.0	Noise 0.1	Noise 0.5	Noise 1.0	Noise 2.0
GPT-2 + APE (128)	4.8	1.2	2.4	2.6	3.5
GPT-2 + APE (256)	17.4	11.9	4.6	3.6	3.2
GPT-2 + APE (512)	14.0	16.3	16.7	17.9	14.3
Meta + APE (128)	98.7	98.6	67.5	55.6	42.8
Meta + APE (256)	81.8	79.7	48.9	43.1	37.9
Meta + APE (512)	100.0	100.0	79.5	65.5	57.1
Meta + RoPE (128)	98.1	100.0	100.0	96.0	88.9
Meta + RoPE (256)	100.0	100.0	100.0	97.9	82.6
Meta + RoPE (512)	100.0	100.0	100.0	98.8	81.0

Table 12. Accuracy (%) on the Parity task under different ablations: zeroing the positional encoding (No Pos), text embeddings (No Embed), or both, only on the meta-token.

Model (Train Length)	Full	No Pos	No Embed	Neither
Meta + APE (128)	100.0	100.0	100.0	100.0
Meta + APE (256)	75.0	77.4	77.4	79.8
Meta + APE (512)	67.9	71.4	72.6	66.7
Meta + RoPE (128)	100.0	97.6	100.0	100.0
Meta + RoPE (256)	66.7	66.7	73.8	66.7
Meta + RoPE (512)	76.2	75.0	75.0	64.3

Table 13. Accuracy (%) on the Copying task under different ablations: zeroing the positional encoding (No Pos), text embeddings (No Embed), or both, only on the meta-token.

Model (Train Length)	Full	No Pos	No Embed	Neither
Meta + APE (128)	96.6	93.2	7.2	4.8
Meta + APE (256)	98.2	99.6	5.0	3.6
Meta + APE (512)	99.0	96.6	5.7	5.4
Meta + RoPE (128)	100.0	99.6	6.9	4.9
Meta + RoPE (256)	100.0	100.0	4.5	5.1
Meta + RoPE (512)	100.0	95.6	6.9	4.9

Table 15. Accuracy (%) on the Copying task with Gaussian noise added to positional encoding.

Model (Train Length)	Noise 0.0	Noise 0.1	Noise 0.5	Noise 1.0	Noise 2.0
GPT-2 Abs (128)	2.9	1.2	0.0	0.0	0.0
GPT-2 Abs (256)	6.0	7.1	3.6	0.8	0.7
GPT-2 Abs (512)	6.0	5.8	3.6	0.4	0.3
Meta + APE (128)	96.1	98.5	69.8	58.6	54.9
Meta + APE (256)	100.0	100.0	76.3	68.8	57.2
Meta + APE (512)	98.9	98.7	74.4	68.9	50.5
Meta + RoPE (128)	100.0	100.0	75.9	68.6	49.9
Meta + RoPE (256)	100.0	100.0	82.6	65.6	45.1
Meta + RoPE (512)	100.0	100.0	84.4	67.6	46.3

H.4. Length Generalization Ability under No Positional Encoding Ablation

Table 16. List-Recall: "No Pos" vs. Full accuracy for Meta-attention with APE and Meta-attention with RoPE.

Model (Split, Train Len)	Full	No Pos	Δ (pp)
Meta + APE (128)			
small	—	—	—
medium	77.8%	88.9%	+11.1
hard	11.1%	22.2%	+11.1
Meta + APE (256)			
small	100.0%	100.0%	0.0
medium	100.0%	100.0%	0.0
hard	44.4%	22.2%	–22.2
Meta + APE (512)			
small	—	—	—
medium	—	—	—
hard	100.0%	100.0%	0.0
Meta + RoPE (128)			
small	—	—	—
medium	44.4%	55.6%	+11.1
hard	11.1%	11.1%	0.0
extra-hard	0.0%	0.0%	0.0
long	0.0%	11.1%	+11.1
Meta + RoPE (256)			
small	100.0%	100.0%	0.0
medium	100.0%	100.0%	0.0%
hard	33.3%	66.7%	+33.3
extra-hard	0.0%	22.2%	+22.2
long	0.0	0.0	0.0
Meta + RoPE (512)			
small	—	—	—
medium	100.0%	100.0%	0.0
hard	100.0%	100.0%	0.0
extra-hard	44.4%	55.6%	+11.1
long	0.0%	0.0%	0.0

I. Examining Context Compression with Rate-Distortion Theory

Given that these results provide evidence that meta-tokens can compress context in their representation, we develop mathematical formalizations to analyze this behavior. In particular, we turn to information-theoretic tools – specifically, an information bottleneck view.

For a meta-token at x_m succeeding a sequence of tokens $X = x_{i:m-1}$ from indices i to $m-1$, we consider a compression function $\zeta(\cdot)$ which transforms the subsequence X into x_m. As such, we define $\hat{X} = \zeta(X) = \zeta(x_{i:m-1})$ to be the *compressed representation stored* in x_m. This can be generalized to the full set of M meta-tokens:

$$\hat{X}_{1:M} = [\zeta_1(X_{1:m_1-1}), \zeta_2(X_{m_1+1:m_2-1}), \dots \zeta_M(m_{M+1} : m_n)]$$

For practicality, we consider the variational information bottleneck (Alemi et al., 2017). This introduces an encoder $q_\phi(\hat{x} \mid x)$ and decoder $q_\theta(y \mid \hat{x})$, along with a simple prior $r(z)$ (e.g. $N(0,1)$), yielding the following form to solve for these variational distributions:

$$\min_{q_\phi, q_\theta} \mathop{\mathbb{E}}_{p(x,y)} [\mathop{\mathbb{E}}_{q_\phi(\hat{x}|x)} [-\log q_\theta(y \mid \hat{x})] + \beta \cdot \mathop{\mathbb{E}}_{p(x)} [KL(q_\phi(\hat{x} \mid x) || r(x))]$$

This form admits an equivalent perspective in rate-distortion theory. Specifically, the first term measures the quality in predicting the downstream target given a lossy compression $\hat{X}$ ("distortion"). The second term measures the average number of bits required to encode $\hat{X}$, relative to some simple reference code $r(z)$ ("rate"). As such, analyzing rate-distortion curves – sweeping over values of β – can provide valuable insights into the quality of the "compression" behavior and its informativeness when the meta-token is attended to.

Theorem I.1. *Let $D_{abs}(R)$ be the minimum distortion achievable at rate R under the VIB objective only using absolute positional encoding (no meta-tokens), and let $D_{meta}(R)$ be the minimum distortion achievable at rate R with meta-tokens. Then, for every $R \geq 0$,*

$$D_{meta}(R) \leq D_{abs}(R) \tag{1}$$

Proof. The meta-tokens are simply a new (latent) channel that may be utilized to search for candidate distributions. However, this latent can be ignored, yielding the original search space; that is, any encoder $q_\phi(\hat{x} \mid x)$ that does not use meta-tokens can be implemented in the meta-token model by zeroing out all meta-token contributions. Therefore, $\mathcal{Q}_{\text{abs}} \subseteq \mathcal{Q}_{\text{meta}}$, where $q = (q_\phi, q_\theta)$ over the feasible combinations of encoder and decoder. Naturally, minimizing a function over a larger feasible set cannot increase its minimum. Thus, for a fixed rate R,

$$D_{\text{meta}}(R) = \min_{q \in \mathcal{Q}_{\text{meta}} :\ I(X;\hat{X})=R} D(q) \ \leq \min_{q \in \mathcal{Q}_{\text{abs}} :\ I(X;\hat{X})=R} D(q) = D_{\text{abs}}(R).$$

Note that the same result holds for RoPE in place of APE (i.e. D_{RoPE} in place of D_{abs}), as well. □

Intuitively, meta-tokens expand the feasible set of encoders and decoders, which will either match or lower distortion for a given rate. Thus, the quality of compression with respect to its informativeness in predicting the target can only improve.

Rate-Distortion Informs the Quality of Context Caching. To obtain empirical rate–distortion curves for our meta-token bottleneck in Figure 3, we freeze the pre-trained meta-token model and fix a small variational bottleneck head to the *last* meta-token hidden state. Concretely, let $h_m \in \mathbb{R}^D$ be the output of the final Transformer layer at the last meta-token position. We introduce

$$q_\phi(z \mid h_m) = \mathcal{N}\big(\mu_\phi(h_m),\, \mathrm{diag}(\sigma_\phi^2(h_m))\big), \quad q_\theta(y \mid z) = \mathrm{softmax}(Wz + b),$$

with $\mu_\phi, \sigma_\phi : \mathbb{R}^D \to \mathbb{R}^L$ and $W \in \mathbb{R}^{|\mathcal{V}| \times L}$. We then optimize the ELBO:

$$\min_{\phi,\theta}\ \mathbb{E}_{h_m,y}\big[-\log q_\theta(y \mid z)\big] \ +\ \beta\, \mathbb{E}_{h_m}\big[\mathrm{KL}\big(q_\phi(z \mid h_m) \,\|\, \mathcal{N}(0, I)\big)\big].$$

Training is performed on the **small** List-Pointer F.2.1 split (50 examples, batch size 1), for 5 epochs at each $\beta \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0\}$. After each run, we record the *average* cross-entropy loss ("distortion") and KL ("rate") on the same 50 examples. Finally, we plot the resulting rate–distortion curves on a `symlog` x-axis (linear below 20 nats, logarithmic above) so that both the low-rate "knee" and the long tail are visible (see Figure 3).

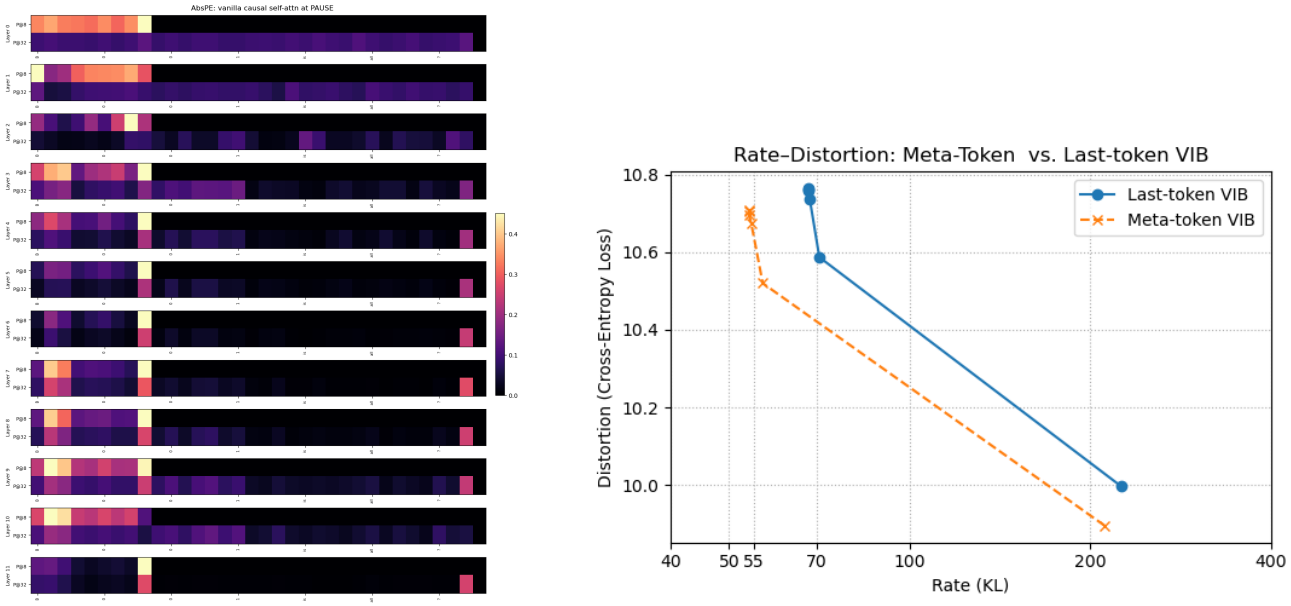


Figure 3. (Left) This plot visualizes the residual stream after each layer, to analyze the meta-token within causal attention. The colors before the meta-token (the colored band across the layers) denote the context which the meta-token attends to and implicitly stores, and the final, rightmost colored line represents the final meta-token in the sequence, which attends to the previous one at the aforementioned band. (Right) We analyze the variational information bottleneck (VIB) objective and its decomposition into its rate and distortion components. Supporting the findings of Theorem 5.1, for a given rate R, the distortion D is strictly lower for the meta-token compared to the last non-meta-token element in the sequence.

J. Theoretical Analysis

J.1. Proof of Theorem 3.1

Lemma J.1. *Let $\ell_1, \ell_2, \ldots, \ell_N$ be logits and define softmax distribution $\alpha_j = \frac{\exp(\ell_j)}{\sum_{k=1}^{N} \exp(\ell_k)}$. Suppose that for some "correct" index j^* we have $\ell_{j^*} = L$, and for all other indices $j \neq j^*$, $\ell_j \leq L - \Delta$ for some $\Delta > 0$. Then, entropy $H(\alpha)$ is strictly decreasing in Δ.*

Proof. First, we can group the other logits (i.e. $j \neq j^*$, such that $S = \sum_{j \neq j^*} e^{\ell_j}$. Then, since each ℓ_j carries the property that $e^{\ell_j} \leq e^{L-\Delta}$ given $\ell_{j^*} = L$, we have that $S \leq (N-1)e^{L-\Delta}$ since there are $N-1$ terms. Revisiting the softmax α, we have that $\alpha_{j^*} = \frac{e^L}{e^L+S} \geq \frac{e^L}{e^L+(N-1)e^{L-\Delta}} = \frac{1}{1+(N-1)e^{-\Delta}}$. We will denote this quantity as p henceforth. Next, each other softmax α_j for $j \neq j^*$ must have the property that $\alpha_j = \frac{e^\ell}{e^L+S} \leq \frac{e^{L-\Delta}}{e^L(1+(N-1)e^{-\Delta})} = \frac{e^{-\Delta}}{1+(N-1)e^{-\Delta}} = \frac{1-p}{N-1}$.

As a result, we have the following entropy maximization problem:

$$
\begin{aligned}
\underset{\alpha_1,\ldots,\alpha_N}{\text{maximize}} \quad & -\sum_{j=1}^{N} \alpha_j \log \alpha_j \\
\text{subject to} \quad & \sum_{j=1}^{N} \alpha_j = 1, \\
& \alpha_{j^*} = p, \\
& \alpha_j \geq 0, \quad j = 1, \ldots, N.
\end{aligned}
$$

Observe that the entropy (objective) function is Schur-concave in α, so it is maximized when $\alpha_{j^*} = p$ and the remaining

softmax mass is split uniformly over the $N-1$ elements, i.e. $\alpha_j = \frac{1-p}{N-1} \, \forall \, j \neq j^*$. Plugging this in for $H(\alpha)$ yields:

$$H(\alpha) \leq -p \log p - (1-p) \log(1-p) + (1-p) \log(N-1) \tag{2}$$

Next, we aim to study the relationship between H and Δ. By the chain rule, $\frac{dH}{d\Delta} = \frac{dH}{dp} \cdot \frac{dp}{d\Delta}$. $\frac{dH}{dp} = -(1 + \log p) + \log \frac{1-p}{N-1} + 1 = \log \frac{1-p}{(N-1)p}$. Substituting $\frac{1-p}{p} = (N-1)e^{-\Delta}$, we get $\frac{dH}{dp} = -\Delta$ and since $\Delta > 0$, $\frac{dH}{dp} < 0$. We then turn to $\frac{dp}{d\Delta} = \frac{(N-1)e^{-\Delta}}{[1+(N-1)e^{-\Delta}]^2} > 0$ since both numerator and denominator must be > 0. Therefore, $\frac{dH}{d\Delta} = -\Delta \frac{(N-1)e^{-\Delta}}{[1+(N-1)e^{-\Delta}]^2} < 0$, meaning that $H(\alpha)$ is strictly decreasing in the margin Δ. □

We will now use Lemma J.1 to prove Theorem 3.1.

Proof of Theorem 3.1. Consider a parametrized path by variable $t \in [0, \Delta]$; define $\ell_j^{(t)} = \ell_j + \delta_{j,j^*} t$, and $\alpha_j^{(t)} = \frac{e^{\ell_j^{(t)}}}{\sum_{k=1}^{N} e^{\ell_k^{(t)}}} = \frac{e^{(\ell_j + \delta_{j,j^*} t)}}{\sum_{k=1}^{N} e^{(\ell_k + \delta_{k,j^*} t)}}$. Define $\ell_j^{'(t)} = \frac{d}{dt} \ell_j^{(t)}$ and $\alpha_j^{'(t)} = \frac{d}{dt} \alpha_j^{(t)}$.

Next, we differentiate the entropy $H(\alpha)$ with respect to t:

$$\frac{d}{dt} H(\alpha) = -\sum_{j=1}^{N} [\alpha_j' \ln \alpha_j + \alpha_j \frac{\alpha_j'}{\alpha_j}] = -\sum_{j=1}^{N} \alpha_j' (1 + \ln \alpha_j) = -\sum_{j=1}^{N} \alpha_j' + \alpha_j' \ln \alpha_j$$

Since $\sum \alpha_j' = 0$ due to $\sum \alpha_j = 1$, this simply reduces to $\frac{d}{dt} H(\alpha) = -\sum_{j=1}^{N} \alpha_j' \ln \alpha_j$.

From (Cover & Thomas, 2006), we have that $\alpha_j' = \alpha_j(\ell_j - \mathbb{E}_\alpha[\ell'])$, where $\mathbb{E}_\alpha[\ell'] = \sum_{k=1}^{N} \alpha_k \ell_k'$. Plugging this into the expression for the derivative of entropy with respect to t:

$$\frac{d}{dt} H(\alpha) = -\sum_j \alpha_j (\ell_j' - \mathbb{E}_\alpha[\ell']) \ln \alpha_j = -(\sum_j a_j \ell_j' \ln \alpha_j - \mathbb{E}_\alpha[\ell'] \sum_j \alpha_j \ln \alpha_j)$$

Observe that $\sum_j \alpha_j \ln \alpha_j = \mathbb{E}_\alpha[\ln \alpha]$ so this simply reduces as:

$$\frac{d}{dt} H(\alpha) = -(\mathbb{E}_\alpha[\ell' \ln \alpha] - \mathbb{E}_\alpha[\ell'] \, \mathbb{E}_\alpha[\ln \alpha]) = -\text{Cov}_\alpha(\ell', \ln \alpha) \tag{3}$$

Revisiting the meta-token setup where only the "correct" logit at j^* is boosted, this suggests that $\ell_j' = \mathbf{1}(j = j^*)$. Therefore, $\mathbb{E}_\alpha[\ell'] = \alpha_{j^*}$ and $\mathbb{E}_\alpha[\ell' \ln \alpha] = \alpha_{j^*} \ln \alpha_{j^*}$. This can be substituted into the covariance term above:

$$\frac{d}{dt} H(\alpha) = -\text{Cov}_\alpha(\ell', \ln \alpha) = -(\alpha_{j^*} \ln \alpha_{j^*} - \alpha_{j^*} \, \mathbb{E}_\alpha[\ln \alpha]) = -\alpha_{j^*} (\ln \alpha_{j^*} - \mathbb{E}_\alpha[\ln \alpha])$$

Due to the Schur-concavity of $H(\alpha)$ (Marshall et al., 2011), $\ln \alpha_{j^*} = \max_j \ln \alpha_j$ and $\ln \alpha_{j^*} > \mathbb{E}_\alpha[\ln \alpha]$. As such, given $\alpha_{j^*} > 0$ and $\ln \alpha_{j^*} - \mathbb{E}_\alpha[\ln \alpha] > 0$, this suggests that $\text{Cov}_\alpha(\ell', \ln \alpha) > 0$ and thus, $\frac{d}{dt} H(\alpha) < 0$. Therefore, we conclude that adding a positive logit boost at the meta-token index ("correct" logit) strictly decreases entropy, supporting the proposed "anchoring" effect notion.

□

J.2. Expressivity of Representable Biases

Theorem J.2. *Consider functions $p : \{0, \ldots, T-1\} \to \mathbb{R}$ and $b : \{-(T-1), \ldots, T-1\} \to \mathbb{R}$ for absolute postional biases and relative biases, respectively. Let $\mathcal{B}_{abs}$ to be the set of all fixed absolute positional bias matrices $B_{i,j}^{abs} = p(j)$ and*

$\mathcal{B}_{rel}$ to be the set of all fixed relative biases $B^{rel}_{i,j} = b(i-j)$. Let $\mathcal{B}_{meta}$ be the set of bias matrices implementable by the Transformer augmented with meta-token embeddings $\{m_t\}$ which emit a content-dependent logit boost at their respective indices. Then,

$$\mathcal{B}_{abs} \cup \mathcal{B}_{rel} \subsetneq \mathcal{B}_{meta} \tag{4}$$

Proof. We break this argument down into two parts → (i.) the forward direction, where we show that all absolute and relative biases without meta-tokens can be modeled by the meta-token model.

(i) $\mathcal{B}_{\text{abs}} \cup \mathcal{B}_{\text{rel}} \subseteq \mathcal{B}_{\text{meta}}$. Every $B \in \mathcal{B}_{\text{meta}}$ is obtained by choosing meta-token embeddings $e_t \in \mathbb{R}^d$ at each position t and a linear head W, so that the total bias at (i, j) is $B_{i,j} = \sum_t Q_i^\top W\, e_t\, \mathbf{1}_{j=t}$.

- *Absolute case.* Given $p(j)$, set $W \in \mathbb{R}^{1\times d}$ and choose each e_j so that $Q_i^\top W\, e_j = p(j)\ \forall\ i$. All other $e_{t\neq j}$ are zero. Then $B_{i,j} = p(j)$.
- *Relative case.* Given $b(i-j)$, place a meta-token at every position j. Choose W and embeddings e_j so that $Q_i^\top W\, e_j = b(i-j)\ \forall\ i, j$.

  For instance, if we let $W = \mathrm{I}d$ and arrange that e_j encodes the vector $\big(b(1-j), b(2-j), \ldots, b(T-j)\big)$, then $Q_i^\top e_j = b(i-j)$ when Q_i is the i-th standard basis vector.

Therefore, every absolute or relative bias (in $\mathcal{B}_{\text{abs}}$ and $\mathcal{B}_{\text{rel}}$) lies in $\mathcal{B}_{\text{meta}}$.

(ii) There exists a bias $B^* \in \mathcal{B}_{\text{meta}}$ such that $B^* \notin \mathcal{B}_{\text{abs}} \cup \mathcal{B}_{\text{rel}}$. Define a content-dependent bias $B^*_{i,j} = f\big(C_j\big)$ where C_j is the full token context preceding position j and f is any non-constant function. Such a B^* arises by setting each meta-token embedding $e_j = f(C_j)$ and $W = \mathrm{I}d$, so $B^* \in \mathcal{B}_{\text{meta}}$.

However, if there was $B^* \in \mathcal{B}_{\text{abs}}$, then there is $p(j)$ with $p(j) = f(C_j)$ for all j and all possible C_j, which is impossible since C_j varies. Furthermore, if $B^* \in \mathcal{B}_{\text{rel}}$, then there is $b(i-j)$ with $b(i-j) = f(C_j)$ independent of i; again, this condition is impossible to be satisfied. Therefore $B^* \notin \mathcal{B}_{\text{abs}} \cup \mathcal{B}_{\text{rel}}$.

As a result, we conclude that the biases represented by $\mathcal{B}_{\text{meta}}$ contain the set of both absolute and relative biases without meta-tokens, and represent additional biases that cannot be represented without meta-tokens. □

The result of Theorem J.2 is that the introduction of meta-tokens strictly grows the expressivity of biases that may be represented, while still being entirely inclusive of the fixed realizable absolute and relative encoding biases. As a result, we do not "lose" anything representationally by introducing meta-tokens, from a positional biases standpoint. This enhanced expressive power also plays a role in enabling the model to learn to focus attention on relevant context spans, reinforcing the aforementioned sharpening effect.

K. Example Prompts for Synthetic Tasks

We provide example prompts and target completions for each synthetic task used in our experiments. Each example illustrates the input format, the placement of _PAUSE_ meta-tokens, and the expected model output.

List Recall

Prompt:

```
Fruits: orange peach banana plum Tools: hammer wrench level pliers
Sports: boxing baseball boxing golf Spices: turmeric cardamom cumin
cinnamon Animals: cat tiger wolf fox Professions: teacher nurse lawyer
architect Vegetables: onion cucumber broccoli cucumber Instruments:
piano clarinet violin piano Professions: teacher nurse lawyer _PAUSE_
architect Q: What is item 3 of Professions? _PAUSE_
```

Completion:
lawyer

Segment Counting

Prompt:
Fruits: orange peach banana plum Tools: _PAUSE_ hammer wrench level pliers _PAUSE_ Sports: boxing baseball boxing golf Spices: turmeric cardamom cumin cinnamon Q: How many times does wrench appear between the pauses around Tools? _PAUSE_

Completion:
1

Parity

Prompt:
Bits: 0 _PAUSE_ 1 0 0 Q: What is the XOR of all bits before this pause? _PAUSE_

Completion:
0

Copying

Prompt:
...as ever Mr _PAUSE_ . Darcy had been, was much relieved by discerning such different feelings.
They had not been long together before Darcy told her that Bingley was also coming to wait on her; and she had barely time to express her satisfaction, and prepare for such a visitor, when Bingley's quick step was heard on the stairs, and in a moment he entered the room. All Elizabeth's anger against him had been long done away; but had she still felt any, it could hardly have stood its ground against the unaffected cordiality with which he expressed himself on seeing her again. He inquired in a friendly, though general, way, after her family, and looked and spoke with the same _PAUSE_ good-humoured ease that he had ever done. Q: Copy the bracketed text. _PAUSE_

Completion:
. Darcy had been, was much relieved by discerning such different feelings.
They had not been long together before Darcy told her that Bingley was also coming to wait on her; and she had barely time to express her satisfaction, and prepare for such a visitor, when Bingley's quick step was heard on the stairs, and in a moment he entered the room. All Elizabeth's anger against him had been long done away; but had she still felt any, it could hardly have stood its ground against the unaffected cordiality with which he expressed himself on seeing her again. He inquired in a friendly, though general, way, after her family, and looked and spoke with the same.

L. Broader Impacts Statement

Our work on learned meta-tokens and meta-attention offers a lightweight, data-efficient way to pre-train language models while demonstrating strong performance when fine-tuned for recall tasks. This suggests a path toward more capable, leaner language models that could be used to handle contexts such as like long legal or medical documents, extended multi-turn dialogues, or large codebases without resorting to prohibitively large architectures or expensive fine-tuning runs. Such models could bring real benefits to areas such as conversational agents for education or healthcare. Building off of prior literature that performs a more explicit learned retrieval from the context (Mohtashami & Jaggi, 2023), this could induce improved and efficient in-line retrieval over vast corpora.

Our work relates strongly to the recent debates in the language modeling community on the impact of positional encoding, particularly around works such as NoPE (Kazemnejad et al., 2023). We provide strong evidence that zeroing the positional encoding can improve performance, providing motivation for hybrid attention mechanisms such as RNoPE (Yang et al., 2025), and other, more efficient ways to pre-train language models with long-context modeling settings in mind. We note that advances in long-context modeling could introduce risks around misuse and unintended harm. More powerful context understanding over long ranges can fuel phishing text and distracted models, especially in the phase of noisy context. However, models trained on corpora without data pre-processing a priori may be subject to harmful behavior such as profane generations. In the context of our work, which uses standard, pre-filtered corpora, this issue is avoided; we encourage users to audit the data used for pre-training first.