Transformer Language Models without Positional Encodings Still Learn Positional Information

Adi Haviv[τ] **Ori Ram**[τ] **Ofir Press**[ω] **Peter Izsak**[ι] **Omer Levy**[τμ]

[τ] Tel Aviv University
[ω] University of Washington
[ι] Intel Labs
[μ] Meta AI

Abstract

Transformers typically require some form of positional encoding, such as positional embeddings, to process natural language sequences. Surprisingly, we find that transformer language models without any explicit positional encoding are still competitive with standard models, and that this phenomenon is robust across different datasets, model sizes, and sequence lengths. Probing experiments reveal that such models acquire an implicit notion of absolute positions throughout the network, effectively compensating for the missing information. We conjecture that causal attention enables the model to infer the number of predecessors that each token can attend to, thereby approximating its absolute position.

1 Introduction

The attention mechanism (Bahdanau et al., 2015) of the transformer (Vaswani et al., 2017) is agnostic to the position and order of tokens in the input sequence. It is therefore common practice to inject positional information via absolute positional embeddings (Vaswani et al., 2017; Radford et al., 2018) or relative bias factors (Shaw et al., 2018; Raffel et al., 2020; Press et al., 2021). In this work, we demonstrate that transformer language models *without* any explicit positional information learn an implicit notion of absolute positions, sufficient for achieving competitive performance.

We compare the performance of language models trained without any explicit positional information (*NoPos* language models) to those trained with three different position-aware mechanisms, namely: sinusoidal embeddings (Vaswani et al., 2017), learned embeddings (Gehring et al., 2017), and ALiBi (Press et al., 2021). Results show that NoPos models are competitive with position-aware models consistently across datasets, model sizes, and input sequence lengths (e.g. Figure 1).

To shed light on our findings, we compare NoPos language models to *relative* and *absolute* position mechanisms via probing. Specifically, we train classifiers to predict the position of a token given its representation across different layers along the network. We find that the NoPos model behaves remarkably similar to a model equipped with learned absolute position embeddings.

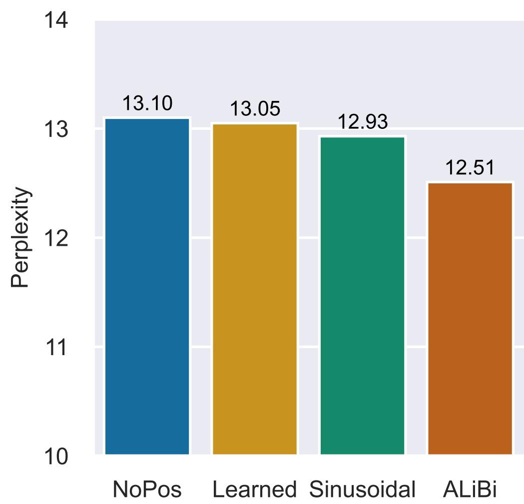

Figure 1: Transformer language models trained without explicitly encoding positional information (*NoPos*) approach the performance of models trained with various positional encoding methods. All models have 1.3B parameters, and are trained on an excerpt of the Pile.

We hypothesize that this surprising behavior is tied to the *causal* attention mechanism, which implicitly injects positional information into self-attention by limiting which tokens are attendable from each position. A model that can count how many tokens precede each position essentially has access to absolute positions. To test our hypothesis, we run similar experiments for masked language models (Devlin et al., 2019), where attention is order-invariant. Indeed, bidirectional models fail to converge when position information is absent, substantiating our hypothesis.

2 Positional Encodings

Transformer models are a stack of interleaved self-attention and feed-forward layers, which are both order-invariant. Hence, to utilize the order of the input tokens, some form of positional information is explicitly introduced into the model. Absolute positions are commonly encoded as vectors (one for each position), which are then added to the input tokens' embeddings and fed to the first layer of the transformer. Relative positions are typically encoded as biases (that are added to attention scores) within the self-attention layers. In this work, we consider three popular methods as baselines:

Learned Embeddings trained for absolute positions (Sukhbaatar et al., 2015; Gehring et al., 2017). Learned positional embeddings are commonly used in masked language models (Devlin et al., 2019; Liu et al., 2019) as well as large autoregressive language models, such as GPT-3 (Brown et al., 2020).

Sinusoidal Constant vectors computed by a deterministic function of the input token's absolute position. Sine and cosine functions of different frequencies are used, such that each dimension of the positional encoding corresponds to a sinusoid. Sinusoidal embeddings were introduced in the original transformer for machine translation (Vaswani et al., 2017), and are also used in language modeling (Baevski and Auli, 2019).

ALiBi Attention with linear biases (Press et al., 2021) injects information about the relative distances between input tokens by adding negative biases to the attention scores, which grow linearly with the distance between each pair of tokens.

3 Experiment Setup

Intuitively, encoding positional information explicitly is crucial for enabling transformers to predict the next token in a sequence. To test this intuition, we compare the performance of models trained from scratch without any explicit positional information (denoted as *NoPos*) to those trained with the various positional encoding methods discussed in Section 2, using validation set perplexity. We investigate the canonical WikiText-103 setting (Merity et al., 2017; Baevski and Auli, 2019), as well as a newer large scale setting based on the Pile corpus (Gao et al., 2020) and model architectures inspired by Brown et al. (2020), where we cover a spectrum of models sizes and sequence lengths.

The Canonical Setting (WikiText-103) The WikiText-103 corpus (Merity et al., 2017) is comprised of over 100 million words extracted from a set of high-quality Wikipedia articles. The corpus is tokenized at the word level, resulting in a vocabulary of over 267K tokens.

For this corpus, we use the adaptive embedding transformer model of Baevski and Auli (2019), which contains 16 transformer layers with 1024 model dimensions, 4096 feed-forward dimensions, and 8 attention heads. We train with their exact optimization hyperparameters, as implemented in `fairseq` (Ott et al., 2019), with the exception of the input sequence length, which is shortened to 512 tokens (instead of 3072), as in Press et al. (2021). Detailed hyperparameters are provided in Appendix A.

The Large Scale Setting (The Pile) The Pile (Gao et al., 2020) is an 800GB English text dataset composed of Common Crawl and 22 other diverse sources. For our experiments, we use 2 out of 30 shards;[1] of these, we filter out the GitHub and DM Mathematics sources, and remove the shortest 1% and longest 1% of examples from each source to reduce noise. We use GPT-2's subword tokenizer (Radford et al., 2019) to convert the text into token sequences over a vocabulary of 50K tokens. We randomly sample a validation set of 2000 documents (2.6M tokens) from the corpus, while the remaining 15M documents (21B tokens) comprise the training set.

The baseline model in this setting follows the 1.3B parameter architecture of Brown et al. (2020), also known as GPT-3 XL: 24 transformer layers with 2048 model dimensions, 8192 feed-forward dimensions, and 32 attention heads. The default input sequence length is 1024 tokens. Detailed hyperparameters are provided in Appendix A.

We also conduct two scaling experiments in this setting. We first scale the model size by experimenting with the small (125M parameters), medium (350M parameters), and large (760M parameters) variants of this architecture. In addition, we evaluate the effect of changing the sequence length from 256 to 2048 tokens using the XL model.

4 Results

Table 1 compares the performance of training language models with different positional encoding

[1]Shards 00 and 01 can be downloaded from: `https://the-eye.eu/public/AI/pile/train/`

	WikiText-103	The Pile
NoPos	20.97	13.10
Learned	20.42	13.05
Sinusoidal	20.16	12.93
ALiBi	19.71	12.51

Table 1: Validation set perplexity of transformer language models trained with various positional encoding methods. The WikiText-103 setting (Merity et al., 2017) uses the model of Baevski and Auli (2019) on sequences of 512 tokens, while the Pile (Gao et al., 2020) uses a more recent 1.3B parameter architecture (Brown et al., 2020) over 1024 token sequences.

methods. We observe that NoPos language models approach the performance of the other models, with gaps of 0.55 (WikiText-103) and 0.05 (the Pile) perplexity from models with *learned* positional embeddings. In the Pile setting, the performance differences between *NoPos*, *Learned*, and *Sinusoidal*, are small both in absolute terms and with respect to their difference with *ALiBi*. In the WikiText-103 setting, the performance gaps are wider, but are still modest with respect to random seed variance.[2] These results strongly suggest that learning transformer language models without explicit positional encoding is indeed possible.

We observe that this conclusion is rather consistent. Table 2 explores the effects of scaling the number of model parameters in the Pile setting. While smaller models benefit from fixed, non-parametric positional encodings (*Sinusoidal* and *ALiBi*) benefit smaller models, these performance gaps diminish as the models scale up in parameters. Table 3 shows the effect of varying the sequence length in the Pile setting. In this experiment, the gaps between *NoPos*, *Learned*, and *Sinusoidal* remain almost constant, while the benefit of using *ALiBi* increases as sequences become longer. Overall, it seems that the transformer's ability to predict the next word without explicit positional encoding is robust to the selection of corpus, model size, and sequence length.[3]

[2]For context, Press et al. (2020) report that training the sinusoidal model with inputs of length 3072 on WikiText-103 with 5 different seeds can result in gaps of up to 0.9 perplexity between runs (0.34 standard deviation).

[3]In fact, concurrent work (Anonymous, 2022) also makes a similar observation in one of their ablation experiments, providing yet another independent experimental setting for which this conclusion holds.

Model Size	125M	350M	760M	1.3B
NoPos	22.15	16.87	14.29	13.10
Learned	22.04	16.84	14.21	13.05
Sinusoidal	21.49	16.58	14.04	12.93
ALiBi	19.94	15.66	13.53	12.51

Table 2: Validation set perplexity on the Pile, as a function of positional encoding method and model size. All models operate on sequences of 1024 tokens. Smaller models benefit from fixed, non-parametric positional encodings (*Sinusoidal* and *ALiBi*), but these performance gaps diminish as the models scale up.

Seq Length	256	512	1024	2048
NoPos	14.98	13.82	13.10	12.87
Learned	14.94	13.77	13.05	12.72
Sinusoidal	14.84	13.66	12.93	12.62
ALiBi	14.65	13.37	12.51	12.06

Table 3: Validation set perplexity on the Pile, as a function of positional encoding method and sequence length. All models have 1.3B parameters. The performance differences between *NoPos*, *Learned*, and *Sinusoidal* are consistently small, while *ALiBi* slowly becomes more beneficial as sequences become longer.

5 Probing Analysis

Do transformer language models learn some form of positional information to compensate for the absence of explicit positional encoding? To answer this question, we probe each layer of our trained models[4] for positional information. Specifically, we train a 2-layer feed-forward ReLU network to predict the absolute position (0 to 1023) of each token, given the representation produced by each layer. Each layer's probe is trained separately (hyperparameters are provided in Appendix A). As a soft accuracy metric, we measure the mean absolute (surface) distance between the probe's prediction and the token's actual position.

Figure 2 shows that even though the NoPos model starts with no positional information at all (on par with the random baseline), it becomes position-aware within four layers, to a degree that it appears more informed than ALiBi about each token's absolute position. By the middle layer, *NoPos* can predict absolute positions more-or-less as well as the model with learned positional embeddings (*Learned*). Finally, we observe that all models shed off a significant amount of positional information in the final layers, in line with the findings of Voita

[4]We use the 1.3B parameter models trained over 1024-token sequences of the Pile (Section 3).

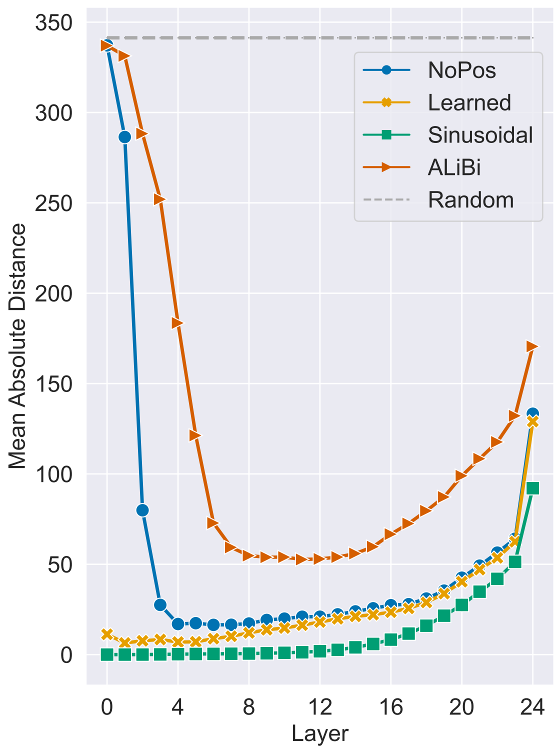

Figure 2: By probing absolute positions, we find that the NoPos model behaves quite similarly to the models embedded with absolute position information (*Learned*). We evaluate the performance using mean absolute distance on 1.3B parameters models trained on the Pile.

0	1	2	3	6	8	11	9
10	10	12	12	9	14	23	18
22	18	13	15	14	18	20	15
18	20	23	30	39	27	29	43
44	61	43	52	46	43	21	33
44	62	43	43	49	43	53	59
43	49	58	62	55	56	57	52
58	58	52	52	58	49	63	63

Figure 3: The absolute position predictions of a probe trained on the 4th layer of a NoPos language model, when given a random 64 token sequence from the Pile's validation set.

	MLM Perplexity
NoPos	147.18
Learned	4.06
Sinusoidal	4.07
ALiBi	4.00

Table 4: Validation set perplexity of *masked* language models (Devlin et al., 2019) trained with various positional encoding methods on an excerpt of the Pile (Gao et al., 2020). The model architecture is based on RoBERTa large (Liu et al., 2019), and processes 128 tokens per sequence. While position-aware models converge to very low perplexities, training without positional encodings (*NoPos*) fails.

et al. (2019). Overall, the probe reveals that models without any explicit positional encoding still learn an implicit notion of absolute positions.

To elucidate what positional information the NoPos model learns, we manually examine a small number of predictions from the layer 4 probe. Figure 3 shows the predictions over a 64 token sequence sampled randomly from the Pile's validation set. We observe that the probe is more accurate at the beginning of the sequence, but becomes fuzzier as it progresses. There appear to be clusters of identical or close numbers, and an overall correlation with the absolute position.

6 Hypothetical Explanation

How do transformers without explicit positional encoding learn absolute positions? We conjecture that the *causal attention* in autoregressive transformer decoders allows them to predict the number of *attendable* tokens at each position, i.e. the number of tokens in the sequence that precede the current one. Such a mechanism could effectively encode the *absolute* position of each token into its vector representation. Indeed, our analysis (Section 5) reveals that some notion of absolute positions exists in the hidden layers of language models, even when they are trained without explicit positional encoding, and that this information is acquired throughout the first few layers.

On the other hand, bidirectional transformer encoders (which are used in masked language modeling, e.g. Devlin et al. 2019) do not contain causal attention masks or any other limitation on the attention mechanism, and thus should not be able to learn absolute positions without explicit positional encoding. We test this corollary by training a masked language model based on RoBERTa large (Liu et al., 2019) in a similar setting to that of the Pile (see Appendix A for hyperparameters). Table 4 shows that, indeed, the NoPos model reaches significantly worse perplexities than the position-informed baselines. This result echoes the findings of Sinha et al. (2021), who also observed that masked language models without positional embeddings suffer significant drops in performance.

7 Conclusion

We find that, contrary to popular belief, transformers language models can and do learn positional information without any explicit positional encoding. Our experiments demonstrate that this phenomenon is robust across different language modeling settings, and that one can approximate the absolute position of each token from the model's internal representations to a surprising degree. However, this phenomenon does not extend to transformer encoders trained on the masked language modeling objective. We conjecture that the causal attention mechanism, which limits attention in one direction of the sequence, is responsible for implicitly imbuing the transformer with positional information.

Acknowledgements

This work was supported by Intel Corporation and Meta Platforms Inc.

References

Anonymous. 2022. What language model to train if you have two million GPU hours? In *Submitted to Challenges & Perspectives in Creating Large Language Models*. Under review.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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 Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Ofir Press, Noah A. Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2996–3005, Online. Association for Computational Linguistics.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *CoRR*, abs/2108.12409.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2440–2448, Cambridge, MA, USA. MIT Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

A Hyperparameters

Table 5 provides the optimization hyperparameters for each one of our experiments, and Table 6 shows the model hyperparameters in the modern (Pile) setting.

	WikiText-103	**The Pile**	**Probe**	**Masked LM**
Sequence Length	512	1024	1024	128
Optimizer	NAG	Adam	Adam	Adam
Peak Learning Rate	1	2e-3	2e-3	1e-3
Warmup Steps	16,000	500	500	500
Total Steps	286,000	10,000	10,000	10,000
Tokens per Batch	72,000	256,000	64,000	1,024,000
Dropout	0.3	0	0	0.1
Weight Decay	0	0.01	0.01	0.01

Table 5: The optimization hyperparameters used in this work. The *NAG* optimizer refers to Nesterov accelerated gradient (Nesterov, 1983), and Adam refers to (Kingma and Ba, 2015).

	125M	**350M**	**760M**	**1.3B**
Layers	12	24	24	24
Model Dimensions	768	1024	1536	2048
Feed-forward Dimensions	3072	4096	6144	8192
Attention Heads	12	16	16	32

Table 6: The models hyperparameters by size.