ExPe: Exact Positional Encodings for Generative Transformer Models with Extrapolating Capabilities

Aleksis Datseris
Graphwise & FMI,
Sofia University St. Kliment Ohridski
datseris@uni-sofia.bg

Sylvia Vassileva
FMI, Sofia University St. Kliment Ohridski
svasileva@fmi.uni-sofia.bg

Ivan Koychev
FMI, Sofia University St. Kliment Ohridski
koychev@fmi.uni-sofia.bg

Svetla Boytcheva
Graphwise & FMI,
Sofia University St. Kliment Ohridski
svetla.boytcheva@graphwise.ai

Abstract

This paper introduces a novel approach to position embeddings in transformer models, named "Exact Positional Embeddings" (ExPE). An absolute positional embedding method that can extrapolate to sequences of lengths longer than the ones it was trained on. Traditional transformer models rely on absolute or relative position embeddings to incorporate positional information into token embeddings, which often struggle with extrapolation to sequences longer than those seen during training. Our proposed method utilizes a novel embedding strategy that encodes exact positional information by overriding specific dimensions of the embedding vectors, thereby enabling a more precise representation of token positions. The proposed approach not only maintains the integrity of the original embeddings but also enhances the model's ability to generalize to longer sequences. In causal language modeling, our ExPE embeddings significantly reduce perplexity compared to rotary and sinusoidal embeddings, when tested on sequences longer than those used in training. The code and supplementary materials can be found in [1]

1 Introduction

The transformer architectures have revolutionized the field of natural language processing (NLP), enabling significant advancements in tasks such as machine translation, summarization, and question answering. Central to the success of transformers is the self-attention mechanism, which allows models to capture complex dependencies between tokens in a sequence. However, a critical challenge in leveraging self-attention is the incorporation of positional information, as the mechanism itself is inherently permutation invariant.

Traditional approaches to position embeddings can be broadly categorized into two main types: absolute and relative methods. Absolute position embeddings, such as those introduced by Vaswani et al. (2017), typically involve adding vectors to the token embeddings. The vector could be fixed or learned, allowing the model to understand the position of each token in the sequence. Although effective, these methods often struggle with generalization to sequences longer than those encountered during training. On the other hand, relative position embeddings, as explored by Shaw et al. (2018) and others, focus on encoding the relative distances between tokens, offering improved flexibility but still facing challenges in capturing precise positional information.

The main issues with the inability to generalize to sequences longer than those seen during training are multi-fold. First, the computational requirements to train a neural network are an order of magnitude larger than the computational resources required for inference with the same neural network. Second, there is an ever-growing need for longer context models. Research has led to the development of models with context lengths of up to 2 million tokens Ding et al. (2024), and that is still not enough for the commercial and research needs we have for those models, since there are many tasks that require more tokens to be solved. Third, and most importantly, is the fact that the primary means by which those models are so effective is the self-attention mechanism, which has quadratic complexity in both time and space. This makes it economically and computationally inefficient for us to scale the context length of those models. One would think that there isn't a fundamental difference in how language is processed and understood as the sequences become longer, or at least the understanding of where the words are in the sequence shouldn't be an issue. What we explore in this pa-

[1] https://github.com/Aleksis99/ExPe/blob/main/Appendices.pdf

Proceedings of Recent Advances in Natural Language Processing, pages 247–253
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-030

per is whether there are ways to give the models positional information in a way that they can generalize to sequences longer than the ones seen during training. If achieved, it could reduce the cost, time, and environmental impact of implementing such models by an order of magnitude.

2 Background and Proposed Approach

Extensive background and related work can be found in supplementary material 1. The attention mechanism Vaswani et al. (2017) is weighted average defined as:

Attention($\mathbf{q}_m, \mathbf{k}_n, \mathbf{v}_n$) = softmax($\frac{\mathbf{q}_m \mathbf{k}_n^T}{\sqrt{d}}$)$\mathbf{v}_n$, where $\mathbf{q}_m = W_q \mathbf{x}_m, \mathbf{k}_n = W_k \mathbf{x}_n$, and $\mathbf{v}_n = W_v \mathbf{x}_n$.

The attention mechanism is permutation invariant, meaning that it has no positional information about where the tokens are in the sequence. Traditional methods for positional encodings either add/apply the information to each embedding or add/apply it to the query ($\mathbf{q}_n = f_q(\mathbf{x}_n)$) and key ($\mathbf{k}_n = f_k(\mathbf{x}_n)$) embeddings.

2.1 Formulation of ExPE

To address the issue with the inability of current approaches to extrapolate to sequences longer than the ones seen during training, we present Exact Positional Encodings (ExPE). In contrast, other approaches attempt to incorporate positional information by augmenting the embedding vector with positional information or manipulating the vector using techniques such as rotations. We propose the idea to override a limited number of values from the vector embedding with a vector representation of the position of the embedding in the sequence.

The main idea we propose is to override l of the embedding dimensions to represent the exact position of the tokens. Starting with an initial value S and increasing by a small constant θ.

$$\begin{aligned} \mathbf{q}_m = f_q(\mathbf{x}_q, m) &= f_q(\varphi(x_q, m)) = \mathbf{W}_q(\varphi(x_q, m)) \\ \mathbf{k}_n = f_k(\mathbf{x}_k, n) &= f_q(\varphi(x_k, n)) = \mathbf{W}_k(\varphi(x_k, n)) \end{aligned} \tag{1}$$

where $\mathbf{W}_q$ and $\mathbf{W}_k$ are the Query and Key matrices of the attention mechanism. The positional encoding ExPE written as $\varphi(x, i)$ is defined as:

$$\begin{aligned} x &:= (x_1, x_2, ..., x_d) \\ \varphi(x, n) &:= (p_n, p_{n+1}, ..., p_{n+l-1}, x_{l+1}, ..., x_d), \\ p_n &:= S + \theta * n \end{aligned} \tag{2}$$

Here S is a defined constant, for example 0, also θ is a predefined constant, for example $1/2m$, d is the dimensionality of the model, and m is the max expected length of input during training. The size of the positional encoding l is also a hyper-parameter whose size can vary. The reason why we are overriding the first l values in the vector instead of concatenating them is that the embeddings are applied before each transformer block, and that way, the size of the embeddings remains the same. We have to note the fact that we override the first l values of the vector, and consider them as a placeholder for the positional information. If token x_n is at position n and token x_m is at position m after applying $\varphi()$ the first l values in the x_n will be $p_n, ..., p_{n+l-1}$ and x_m will be $p_m, ..., p_{m+l-1}$. The difference between $p_n - p_m = \theta * (n - m)$, so the greater the distance between two tokens, the greater the difference in the positional embedding values will increase proportionally.

Since the fundamental principle behind this is that larger values should be interpreted as that the token is further from the start into the input sequence, not only exact and relative information about the positions of the token can be represented in this way, but it should allow the model to extrapolate for sequences that are longer than the inputs the model was trained on something other positional encodings fail to achieve. A visualization to

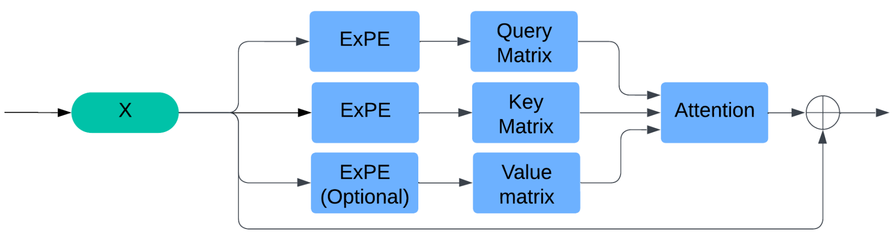

Figure 1: Flowchart showing when ExPE is applied in self-attention.

show better when exactly ExPE is applied to the query and keys is shown in Figure 1. ExPE gets applied to the input of the Query Matrix and Key Matrix since those are the only embeddings that need positional information between them as the matrix multiplication between them will represent the attention between each pair of tokens in the sequence $QK^T \in \mathbb{R}^{n \times n}$ the application to the input of the Value matrix is optional. The Query and Key embedding with encoded positional information, with the Value embeddings, go through the Scaled-Dot-Product attention operation, followed by the residual connection. The application of ExPE before the Query, Key, and Value matrices relies on the ability of those matrices to learn how to use ExPE's representation of positional information. Thanks to the residual connection, the information that ExPE overrides is not entirely lost, as it still travels through it.

2.2 Rationale Behind ExPE

The application of ExPE before the Query, Key, and Value matrices in the Attention relies on the ability of the Attention to learn how to use ExPE's representation of positional information. Thanks to the residual connection, the information that ExPE overrides is not fully lost as it still travels through it.

An intuitive way to think about it is to believe that the model will try to learn to hold the context-independent information in the first part of the vector and the context-dependent information in the second part, which ExPE does not override. For example if think about the word *"dog"* a dog could mean a German Shepherd an American Pit Bull Terrier or even a toy. Even though there is a big difference between the breeds, which are living, breathing creatures, and the toy, there is some context-independent information that unites all of those concepts, which the word *"dog"* could represent in our mind. This context-independent information does not need to be contextualized, so if it gets overridden by ExPE, it will not be lost, as shown before, and it can still change thanks to the contextualized information coming from the attention mechanism.

2.3 ExQPE Quantization Stable Alternative

In fp32 there are about $1e9$ numbers between $[0, 1]$, in fp16 it's $33e6$, in TF32 this drops to about $1e6$ and in bf16 there are only around $16e3$. Which could be problematic when needing a longer context and using heavy quantization. To deal with this, we developed a second version of ExPE called ExQPE, Exact quantizable positional encodings. The idea behind ExQPE is to increment only one of the l dimensions of the positional encodings.

$$\begin{aligned} \mathbf{q}_m &= f_q(\mathbf{x}_q, m) &&= f_q(\varphi(x_q, m)) = \mathbf{W}_q(\varphi(x_q, m)) \\ \mathbf{k}_n &= f_k(\mathbf{x}_k, n) &&= f_q(\varphi(x_k, n)) = \mathbf{W}_k(\varphi(x_k, n)) \end{aligned} \tag{3}$$

where S, θ_1, θ_2 are hyperparameters and

$$\begin{aligned} & x := (x_1, x_2, ..., x_d) \\ & \varphi(x, n) := (p_{n,0}, p_{n,1}, ..., p_{n,l-1}, x_l, ..., x_d), \\ & p_0 := (S + 0 * \theta_1 + \theta_2, \\ & S + 1 * \theta_1, ..., S + (l-1) * \theta 1) \\ & p_{0,i,i \neq 0} := S + i * \theta 1 \\ & p_{k,i} := p_k[i] \\ & p_i := (p_{i-1,0}, p_{i-1,1}, ..., p_{i-1,k-1}, p_{i-1,k} + \theta_2, \\ & p_{i-1,k+1}, ..., p_{i-1,l-1}) \; (where\ k \equiv i \mod l) \end{aligned} \tag{4}$$

3 Data

For the proper training of a language model, high-quality text data is necessary. There can't be specified an exact amount of data, but based on Hoffmann et al. (2022), at least 20 tokens per parameter offer a good compute-to-performance ratio.

Data for Causal Language Modelling and Masked Language Modelling The data used for the initial experiments on causal language modeling and masked language is a subset taken from the Fineweb dataset Penedo et al. (2024). The FineWeb dataset comprises over 15 trillion tokens of cleaned and deduplicated English web data from CommonCrawl. From that dataset, a subset was selected that contained only text at least 4000 tokens long, until a total of 700 million tokens were accumulated. The texts are very complex and vary widely in topics. The llama Touvron et al. (2023) tokenizer was used for tokenization. The total data collected was 700M tokens, of which 100M was equally divided into the dev/test split. The average length of the individual texts was 8000 tokens. The train, dev, and test data split for the experiments are $600,000,000$ training tokens and $50,000,000$ val/test, all with an average length of $8,000$

Data for Bigger Causal Language Models For developing a bigger GPT model, we decided to use a more standard pertaining scheme. We used the Fineweb-edu dataset Lozhkov et al. (2024), a subset of 1.3T tokens from educational web pages filtered from the FineWeb dataset. A random subset of 10 billion parameters was selected from the dataset. The average length of the texts was 1000 tokens and the train, dev, and test data split for the experiments are $600,000,000$ training tokens and $50,000,000$ val/test, all with an average length of $8,000$.

4 Experiments and Results

4.1 Causal Language Modeling

For comparison of out method we will focus mostly on RoPE Su et al. (2023) as they are the state-of-the-art technique for positional encodings, but for historic reasons, we will do some evaluation on the Sinosoidal positional encodings Vaswani et al. (2017) to reevaluate that RoPE is a superior technique.

Experimental Setup. The data used is described in section 3. The hardware used for training was a single A40 48GB GPU. The models were trained on sequences of 512 tokens due to resource limitations.

Both models had 35M parameters. The parameters were based on the parameters of the GPT models from Brown et al. (2020). Two models were compared: one standard decoder transformer model with Sinusoidal PE, one with rotary embeddings, and one that uses the proposed positional embeddings called ExPE for short.

Results. The experiments with this approach yielded a good result (Table 3) compared to RoPE, and the model's cross-entropy also remained stable when given inputs that were 2 or 4 times longer than the training data text, which was 512 tokens in length. This is something that relative, rotary, and sinusoidal positional encoding don't achieve. Our model showed even a slight improvement when the input length was increased. Our explanation for that is that the texts are initially at least 4000 tokens, and the majority of input sequences that we test the model with are cut from a longer text. There is a lot of missing context, as our model can extrapolate positions for more extended sequences. We not only don't see the perplexity increase, but it even decreases slightly.

Ablation Studies on ExPE. We did a series of ablation studies to verify that all aspects of the ExPE positional encodings are necessary. The results can be seen in Table 2. It is clear that all aspects of ExPE are necessary for performance. Increasing the value in the vector we override seems to give a significant performance improvement. This could also be to a large extent because initializing two values in embedding with the same value makes them learn the same thing due to the way the gradient flows through (Kumar, 2017).

With $l = 1$ we achieve significantly worse results. The most significant factor should be simply the fact that the positional information is difficult to distinguish from the other embedding details when it is in a single dimension.

Similarly to how RoPe needs to be used in each transformer block, the same applies to ExPE. It appears that the model struggles to retain the positional information in the embeddings after they are passed through the layers. Therefore, for the model to utilize the position information, it must be applied in each transformer block.

We tried to set S and θ as learnable parameters. Here we see that the model fails to learn basic language modeling. It seems that having only two learned parameters affects so much of the output, making the model unstable and making it difficult for the model to learn.

In (Gu and Dao, 2024) used a specific initialization was used to initialize their model in a stable state. We decide to make S and θ learnable, but instead of initializing them randomly, initialize them with the same values as the ones in the non learned variant of the model. While achieving better results compared to the previous method, the training time was over 60%.

4.2 Large-scale Experiments

The data used is described in section 3. We tokenized the dataset using Mistralai's Mistral-Small-Instruct-2409 Jiang et al. (2023) tokenizer. The architecture used was the Llama 3 architecture without the key-value cache, since it only increases inference speed. The embedding weights of the medium models are shared with their linear units.

The model parameters found in the supplementary material 1 are based on the GPT parameters from Brown et al. (2020).

The results are shown in Table 1. What we see here is that ExPE still manages to maintain its length extrapolation capabilities while achieving comparable performance to RoPE, even at lengths seen during training. While also requiring significantly fewer computational resources.

We also evaluated the performance of the models on the HellaSwag Zellers et al. (2019),

Model	Loss (ev=1)	Loss (ev=2)	Loss (ev=4)	Training time
LLama Small	2.89	3.80	4.95	1.00
ExPE Small	2.87	**2.83**	**3.25**	**0.93**
ExQPE Small	**2.86**	**2.83**	3.35	**0.93**
LLama Medium	**2.63**	3.45	4.55	1.00
ExPE Medium	**2.63**	**2.59**	2.71	**0.78**
ExQPE Medium	**2.63**	**2.59**	**2.68**	**0.78**

Table 1: The legend (ev=1) is the evaluation on texts of length the training length; here, for all models, the training length is 512 tokens. (ev=2) is the evaluation on texts with length double the training length, etc.

Model	Loss (ev=1)	Loss (ev=2)	Loss (ev=4)	Epoch time
ExPE	3.93	3.87	4.26	**1**
ExPE (p is stable)	4.26	4.21	**4.24**	**1**
ExPE (l = 1)	4.48	4.83	6.01	**1**
ExPE (learned initialized)	**3.88**	**3.82**	4.89	1.6
ExPE (learned)	7.33	7.34	7.61	1.6
ExPE (once)	4.43	4.82	6.18	**1**

Table 2: Ablation studies done on ExPE. Where ExPE (p is stable) means we override with $p_i, p_i, .., p_i$ instead of $p_i, p_{i+1}, .., p_{i+l}$. ExPE ($l = 1$) uses $l = 1$ for it's positional encodings. ExPE (once) has its positional encodings applied only once before the first transformer block. ExPE (learned initialized) has its positional encodings S and θ parameters learned but initialized with the same values as the non learned variant. ExPE (learned) has its positional encodings S and θ parameters learned.

	Sinusoidal	RoPe	ExPE
Loss (ev=1)	4.0	**3.88**	3.93
Loss (ev=2)	4.75	4.37	**3.87**
Loss (ev=4)	5.64	5.05	**3.88**

Table 3: The legend (ev=1) is the evaluation on texts with length the training length; here, for all models, the training length is 512 tokens. (ev=2) is the evaluation on texts with length double the training length, etc.

MMLU(Measuring Massive Multitask Language Understanding) Hendrycks et al. (2021), ARC and ARC easy Moskvichev et al. (2023) benchmarks to test how they perform on standard LLM benchmarks and we see that their results (found in Table 5) are comparable. In terms of training inference speed and memory requirements, ExPE is practically equivalent to Sinusoidal, as shown by Press et al. (2022); they are significantly faster than RoPE. In general, even if ExPE doesn't demonstrate a remarkable ability to extrapolate to sequences longer than those seen during training, it still shows significant improvement compared to RoPe and Sinusoidal. Also manages to maintain performance compared to RoPe while requiring significantly less compute when compared to sequence lengths seen during training.

Scaling the Encoding. Chen et al. (2023) showed that if a model uses RoPe at a certain frequency and a certain context length, that model context length can be easily extended by scaling the frequency. If you would like to double the context length, you could scale the frequency of RoPe by half and do additional training. The method is effective because it requires a small amount of training for the model to extrapolate to the new training length. The issue is that you still need to do additional training. We decided to check how well ExPE behaves when scaling the encoding values without additional training. When we trained the model with a context length of 512 tokens, the first positions of ExPE values ranged from 0 to 0.25 while when texting to lengths of 2048 tokens, those values ranged from 0 to 1, so we decided to scale them by half to see how the model behaves without additional training.

The results shown in tables 4 and 5 show that for the longest extrapolation, the scaling helps, but for lengths seen during training, it slightly decreases

Model	Loss (ev=1)	Loss (ev=2)	Loss (ev=4)	Loss (ev=8)	Loss (ev=16)
ExPE M	**2.63**	**2.59**	2.71	5.19	7.56
ExPE M scaled by 0.5	2.76	2.71	<u>2.68</u>	<u>2.98</u>	<u>5.65</u>
ExQPE M	**2.63**	**2.59**	2.68	5.01	7.39
ExQPE M scaled by 0.5	2.74	2.69	**2.65**	**2.76**	**4.98**

Table 4: All results are for ExPE Medium-sized models. The legend (ev=1) is the evaluation on texts of length equal to the training length. Here, for all models, the training length is 512 tokens. (ev=2) is the evaluation on texts with length double the training length, etc.

Model	HS	MMLU	ARC	A E
LLama S	**0.29**	**0.26**	**0.24**	**0.30**
ExPE S	0.28	**0.26**	**0.24**	0.29
ExQPE S	0.27	**0.26**	**0.24**	0.29
LLama M	0.31	**0.26**	**0.26**	0.30
ExPE M	0.31	**0.26**	0.25	0.30
ExPE M (0.5)	0.31	**0.26**	**0.25**	0.30
ExQPE M	**0.32**	**0.26**	0.25	**0.31**
ExQPE M (0.5)	**0.32**	**0.26**	**0.25**	**0.31**

Table 5: Performance of the small (S) and medium (M) sized models on HellaSwag (HS), MMLU, ARC, ARC Easy (A E). The notation <Model (x)> means that the model's encodings are scaled by a factor of x.

performance, and interestingly, for lengths twice the training length, we see the same. This suggests that scaling could be used to enhance performance for more extended periods without requiring additional training. For even longer extrapolations, further training with scaling on longer context lengths should also yield good results with minimal extra training.

5 Limitations

Due to technical limitations, the models trained in this work are too small and were trained on a too short context length, making it difficult for us to reliably state that ExPE and ExQPE are viable state-of-the-art techniques for positional encodings in transformers. Additionally, here we only compare the results of models which have gone through only the pretraining stage and have not gone through the instruction tuning phase with supervised fine-tuning, DPO Rafailov et al. (2024) or Reinforcement learning from human feedback(RLHF) Ouyang et al. (2022). Finally, a general issue with longer context is that human language is inherently local due to the limitations of the human brain; a person can only follow a few sentences at a time. Therefore, many long documents lack long dependencies Yang et al. (2025).

6 Conclusion

In this work, we introduced Exact Position Embeddings (ExPE) and ExQPE, two novel positional encoding methods designed to improve the extrapolation capabilities of transformer models. By explicitly encoding position information precisely, ExPE and ExQPE enhance the model's capabilities for length extrapolation of sequences longer than those seen during training. Our experiments demonstrated that ExPE and ExQPE outperform traditional sinusoidal embeddings and achieve competitive results compared to Rotary Position Embeddings (RoPE), while being more efficient.

We demonstrated that ExPE and ExQPE effectively retain position information and adapt to longer sequences without requiring additional training for the causal language modeling task. We also demonstrated that, with fixed scaling of the two approaches, we can further enhance the length extrapolation capabilities of the model without additional training. Our results indicate that ExPE and ExQPE present a promising alternative to existing positional encoding techniques.

Directions for Future Development For ExPE and ExQPE to be proven as practical techniques for positional encodings, larger experiments should be conducted with more extensive models, greater data, and longer context lengths. The development of proper benchmarks and training data tailored explicitly to long context dependencies is necessary for us to properly test the extended context capabilities of language models. Comparing the method to techniques like Dual Chunk Attention An et al. (2024) and Position Interpolation Chen et al. (2023) should be done.

Acknowledgments

This work is partially supported by the project UNITe BG16RFPR002-1.014-0004 funded by PRIDST.

We are extremely grateful to Prof. DSc. Stoyan Mihov, Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, for giving us initial guidance and helping us define the task at hand when developing this method.

References

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, et al. 2024. Training-free long-context scaling of large language models.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. 2020. Language models are few-shot learners.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, et al. 2024. Longrope: Extending llm context window beyond 2 million tokens.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, et al. 2021. Measuring massive multitask language understanding.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, et al. 2022. Training compute-optimal large language models.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, et al. 2023. Mistral 7b.

Siddharth Krishna Kumar. 2017. On weight initialization in deep neural networks.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. 2023. The conceptarc benchmark: Evaluating understanding and generalization in the arc domain.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, et al. 2022. Training language models to follow instructions with human feedback.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, et al. 2024. Direct preference optimization: Your language model is secretly a reward model.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL-HLT*.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, et al. 2023. Llama: Open and efficient foundation language models.

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.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, et al. 2025. Qwen2.5-1m technical report.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence?