Positional Encoding: Past, Present, and Future

Christopher Curtis - Civic AI Lab (Saiph Savage)

Special Thanks

The Importance of Transformers

Transformers power nearly every state-of-the-art model in:

- NLP
- Vision
- Multimodal Reasoning
- Scientific ML
- Agentic AI
- etc…

Their core innovation, **self-attention**, models global relationships **without recurrence**.

Core Engine: Self-Attention

Transformers have become the backbone of all major modern models because Self-Attention provides:

Global Context Understanding – Every token can attend to every other token ‘simultaneously’.

Parallelization – Removes sequential recurrence, enabling training at scale.

Flexibility Across Domains – NLP, Vision, Audio, Time-Series, Biology, etc.

Scalability – Empirically, performance improves with model and data size.

Where Transformers Excel

- Large Language Models (LLaMA, GPT, Qwen)
- Vision Transformers (ViT)
- Multimodal models (CLIP, Flamingo)
- Information Retrieval, Recommendation Systems (BERT)
- Sentiment Classifiers… etc.

Transformers are powerful: ***but they have a fundamental blind spot***.

The Core Problem: Self-Attention Has No Sense of Order

The transformer models cannot natively detect **adjacency, directionality, or distance**… as we shall soon see.

Sequence data (language, time, DNA, motion) cannot be understood correctly unless order information is provided externally.

This creates the need for **Positional Encoding**:

A means of integrating these concepts into the model.

Why Positional Encoding Is Essential

1) Without PE, Transformers Cannot:

Distinguish “the cat chased the dog” from “the dog chased the cat”

2) Model temporal dependencies
3) Capture distance (how far tokens are from one another)
4) Understand direction (left/right, earlier/later, upstream/downstream)

Positional Encoding integrates ordering, distance, and sometimes directionality into the model pipeline so attention can meaningfully compare tokens.

Positional Encoding - Eras

Positional Encoding has already gone through several ‘eras’ since 2017…

Initial models effectively applied a signal to the model which would encode the absolute position of tokens and left the rest up to the learning process

[2018-2021]

This focus shifted towards prioritizing relative distances instead, and surgically applying learnable parameters to expressions directly inside the attention block

Positional Encoding - Eras

Positional Encoding has already gone through several ‘eras’ since 2017…

[2021-2025]

Currently, there is a split between the two families of transformers (Encoder vs Decoder) with Decoder Transformers converging towards minimal, parameter free methods such as RoPE; and Encoder transformers remaining highly diverse.

[2020-2025]

Multimodal and Domain-Specific schemes began to emerge in 2020, and have since exploded in these past few years.

Why Positional Encoding Is Still Evolving

Domain-specific tasks benefit domain-specific signals

Biology, physics, tabular, and graph data require unique structural cues

Not all tasks share the same “idea of position”

PE is becoming a framework capable of injecting structure into attention

Why This Tutorial Matters

Positional Encoding is essential but often misunderstood, and less accessible

Understanding PE helps you:

- Choose the right architecture
- Diagnose model failures
- Improve extrapolation
- Design domain-specific Transformers

This tutorial aims to equip you with a coherent conceptual toolkit

Why This Tutorial Matters (Soapbox)

Finally, solving a problem in the world of Deep Learning rarely means the solution to one problem alone

As we will see, the evolution of Positional Encoding is often pushed by the needs of quite different research questions…

And yet the solutions to which generalize quite well across domains

Positional Encoding offers a conceptual evolution that is interesting in its own right, but also reveals many things about both neural architecture and mathematical learning practices at the same time

Tutorial Information

Structure of the Tutorial

1. Self-Attention Review
2. Sinusoidal Encoding Deep-Dive
3. Evolution of Positional Encoding Methods
    a. 2018-2021: Conceptual Evolution of RPR Method
    b. 2021-2025 + Multi-Modal/Domain-Specific
4. Common Mathematical Methods
5. What is the future?
6. Short Soapbox

Disclaimer: Code is for Educational Purposes

Formulations shown and **implementations** provided are for **educational purposes** only! These should not be taken as a practical solution.

Sometimes a data structure or formulation deviates slightly from the author's implementation - not in mathematical correctness - but in computational efficiency.

The primary aim here is conceptual and mathematical clarity.

Be wary of directly applying the implementations in practice!

Disclaimer: Some Familiarity is Assumed

We do assume some basic familiarity with Transformers

We will provide a brief review of the minimum requirements to follow along…

However, those who already have a basic understanding of attention mechanisms will benefit the most from this tutorial.

Self Attention Review

Terms and Definitions

Variable Name	Symbol
Batch Size	$B \in \mathbb{N}$
Number of Tokens	$N \in \mathbb{N}$
Maximum Context Length	$N_{max} \in \mathbb{N}$
Embedding Dimension	$D \in \mathbb{N}$
Number of Heads	$H \in \mathbb{N}$
Input	$X \in \mathbb{R}^{B \times N \times D}$
Query, Key, Value Parameters	$W_q, W_k, W_v \in \mathbb{R}^{D \times D}$
Queries, Keys, Values	$Q, K, V \in \mathbb{R}^{B \times N \times D}$
Attention Scores	$\omega \in \mathbb{R}^{B \times N \times N}$
Attention Weights	$\alpha \in \mathbb{R}^{B \times N \times N}$
Context Tensor	$Z \in \mathbb{R}^{B \times N \times D}$
Activation Function	ϕ

Most Important:

N - # of Tokens

D - Embedding Dimension

X - Input

w - Attention Scores

B - Batch Size (IGNORE)

Bonus Slide: What Is a Tensor?

A tensor is just a way of organizing numbers.

It's a **container**: like a list, grid, or cube, filled with numbers the model uses to store information.

If you've worked with:

- A number → that's a 0-D tensor
- A list → that's a 1-D tensor
- A table or grid → that's a 2-D tensor
- A stack of grids → that's a 3-D tensor

You already understand the idea.

Why Do We Need Tensors?

Transformers process many numbers at once:

- batches of sentences
- sequences of tokens
- embeddings with hundreds of dimensions
- multiple attention heads

Tensors let us store all these numbers in neatly structured shapes so the model can perform math efficiently.

Simple Analogy

Think of tensors like boxes:

- A scalar (just a number) is a single number
- A vector (1-D tensor) is a list of numbers
- A matrix (2-D tensor) is a grid of numbers arranged in rows and columns
- A 3-D tensor is a stack of matrices
- A 4-D+ tensor is stacking these in bigger containers

Bonus Slide: What Is an Embedding?

What Is an Embedding?

An embedding is a way for a computer to turn something into numbers so it can understand and work with it.

Humans understand words, images, sounds, sentences: but computers only understand numbers.

An embedding is just a smart list of numbers that represents the meaning of something.

Embeddings are FIXED!

Why Do We Need Embeddings?

Think of an embedding as a “meaning map”:

Words with similar meaning get similar numbers

“cat” and “dog” end up close together

Words with different meaning get very different numbers

“cat” and “pizza” are far apart

Instead of treating each word as totally unrelated, embeddings let the model understand relationships.

Bonus Slide: What Is an Embedding? Example

cat → [0.13, 1.92, -0.45, 0.88, …]

dog → [0.11, 2.01, -0.50, 0.91, …]

pizza → [-1.22, 0.45, 3.88, 0.12, …]

“cat” and “dog” end up close together

“cat” and “pizza” are far apart

Self Attention

$$X \in \mathbb{R}^{B \times N \times D}$$

$$W_Q, W_K, W_V \in \mathbb{R}^{D \times D}$$

$$Q = XW_Q, K = XW_K, V = XW_V \in \mathbb{R}^{B \times N \times D}$$

$$\omega = QK^{T_{1,2}} \in \mathbb{R}^{B \times N \times N}$$

$$\alpha = Softmax(\omega/\sqrt{d}) \in \mathbb{R}^{B \times N \times N}$$

$$Z = \alpha V \in \mathbb{R}^{B \times N \times D}$$

For the purposes of our talk, we will essentially be only concerned with these parts of the Self-Attention Mechanism:

Transformations of input X

Weight Matrices

Attention Scores

Step by Step:

$$X \in \mathbb{R}^{B \times N \times D}$$

$$W_Q, W_K, W_V \in \mathbb{R}^{D \times D}$$

X represents the input, of N tokens, each mapped into a D-dimension embedding.

Wq, Wk, and Wv are all DxD tensors which are used to provide a linear set of weights to operate upon the input in the embedding space.

> Required for the learning operation as the embedding for X is fixed

Note: The shape of our DxD parameter tensors may change in decoder-style architectures, depending on their implementation

Step by Step:

$$Q = XW_Q, K = XW_K, V = XW_V \in \mathbb{R}^{B \times N \times D}$$

This step forms the Query (Q), Key (K), and Value (V) Tensors which are essential throughout of the rest of the attention process.

For the purposes of our talk, we are mostly concerned with the Q and K tensors.

What is inside the Q,K,V's?

$$Q_{[i.j]} = (XW_q)_{[i,j]}$$

Each Q,K,V tensor is in NxD space.

> Each token, and their embedded representation

We can broadly interpret Q,K,Vs as enriched, learnable, encodings of the original input in our embedding space.

What is inside the Q,K,V's?

$$Q_{[i.j]} = (XW_q)_{[i,j]} = \sum_{d=1}^{D} X_{[i,d]} W_{q[d,j]} = X_{[i]} \cdot W_q^T{}_{[j]}$$

More concretely…

Each i,j cell of these tensors (token i, dimension j) is the dot product of the ith token and jth column of weighted matrix across the embedding dimension.

Interpretation: Attention Scores

$$X \in \mathbb{R}^{B \times N \times D}$$

$$W_Q, W_K, W_V \in \mathbb{R}^{D \times D}$$

$$Q = XW_Q, K = XW_K, V = XW_V \in \mathbb{R}^{B \times N \times D}$$

$$\omega = QK^{T_{1,2}} \in \mathbb{R}^{B \times N \times N}$$

$$\alpha = Softmax(\omega/\sqrt{d}) \in \mathbb{R}^{B \times N \times N}$$

$$Z = \alpha V \in \mathbb{R}^{B \times N \times D}$$

What we see in the highlighted equation is at the heart of self-attention.

Interpretation: $\omega = QK^T = XW_q(XW_k)^T$

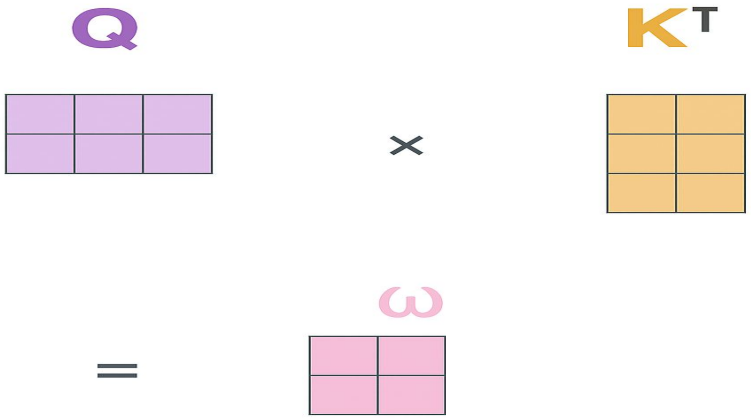
The computation of attention scores serves several high level roles within the attention block itself, perhaps most importantly:

1. It defines the roles for the Query and Key tensors
2. It transforms the tensor’s shape into an NxN
3. Defines the attention values used to form the context tensor later on

Interpretation: $\omega = QK^T = XW_q(XW_k)^T$

Simplistically, we can really think of these scores as answering a key question:

‘Quantitatively, how relevant is each token to another?’



The self-attention calculation in matrix form

This operation yields an NxN matrix with:

- Each row and column corresponding to a token
- Each intersecting cell defining a score which reflects their relationship

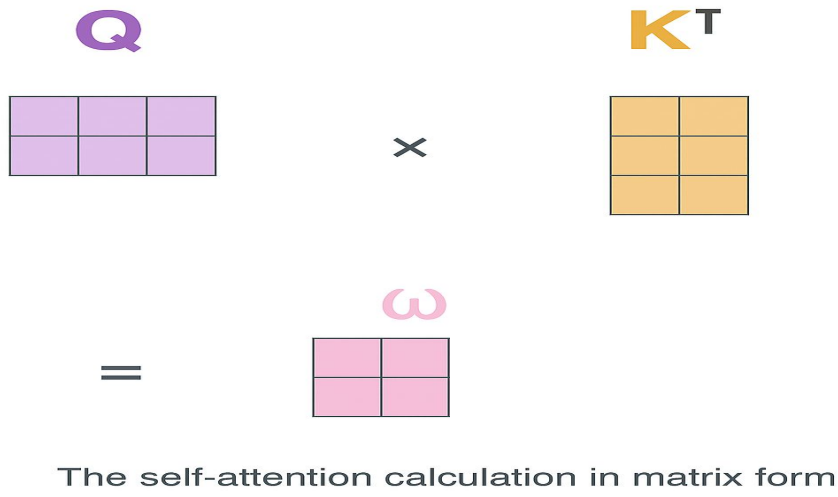
Interpretation: $\omega = QK^T = XW_q(XW_k)^T$

Below we see the expansion of our above equation.

This relationship defines precisely how two tokens, i & j, related to another.

It is also why self-attention is also sometimes referred to as ‘dot-product attention’

The value itself encodes a learnable, sum of each token pair (i,j) accounting for the entire embedding space in each case.



The self-attention calculation in matrix form

$$\omega_{[i,j]} = (QK^T)_{[i,j]} = \sum_{d=1}^{D} Q_{[i,d]} K^T_{[d,j]} = Q_{[i]} \cdot K_{[j]}$$

Bonus Slide: Why the Dot Product?

The **dot-product is also a measure of similarity**:

- When the result is larger, the vectors are more similar
- When the result is smaller, the vectors are less similar

Here similarity refers to pointing in similar directions in the embedding space - In an embedding space, "semantic similarity" is represented by both geometric proximity and alignment.

Note - for those interested (because the dot-product is also magnitude sensitive):

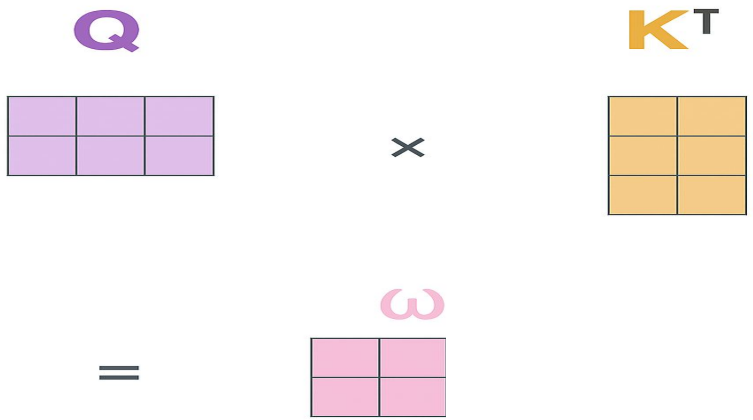
- The use of the Dot Product here motivates scaling later on
- This interaction plays a key role in how embedding choices affect attention

Interpretation: $\omega = QK^T = XW_q(XW_k)^T$

Defines the roles between our Q and K tensors.

With a token pair (i,j) being analogously represented as token i ‘querying’ the column of ‘keys’ for it’s relationship with target token ‘j’.

The model thus learns different relationships for how a token ‘queries’ meaning from other tokens (Q), and how it is ‘looked up’ from others (K).



The self-attention calculation in matrix form

$$\omega_{[i,j]} = (QK^T)_{[i,j]} = \sum_{d=1}^{D} Q_{[i,d]} K^T_{[d,j]} = Q_{[i]} \cdot K_{[j]}$$

Implications

There is no accounting for **ordering** in these expressions!

Attention, by itself, does NOT directly encode any notion of **distance**, **directionality**, or **position** of tokens when computing attention scores.

$$\omega_{[i,j]} = (QK^T)_{[i,j]} = \sum_{d=1}^{D} Q_{[i,d]} K^T_{[d,j]} = Q_{[i]} \cdot K_{[j]}$$

Implications

This creates a need for the introduction of expressions which will reflect some of these concepts during the pipeline…

Typically either in the attention operation itself, or applied directly to the input.

$$\omega_{[i,j]} = (QK^T)_{[i,j]} = \sum_{d=1}^{D} Q_{[i,d]} K^T_{[d,j]} = Q_{[i]} \cdot K_{[j]}$$

Hence the sub-field of Positional Encoding

What is Positional Encoding?

Positional Encoding mechanisms embed token position information to allow models to reason about sequence order.

This often takes the form of boosting or dampening the attention scores of tokens according to their distance and/or some learned relationship.

However, as we will later see, this relationship can also be fixed as well if motivated properly.

Why this topic?

Positional Encoding is a foundational yet often opaque component of Transformer architectures, underpinning how self-attention mechanisms capture sequence order in language, vision, and multimodal models.

Despite its centrality to the success of modern LLMs, and other attention-reliant architectures, the mathematical intuition behind positional encoding remains challenging and inaccessible to many students, researchers and practitioners.

Historical Trajectory of Positional Encoding

Starting at the “Beginning”: $X = U + P$

The first common applications of Positional Encoding were applied directly to the input of the model.

We can formulate this using the above equation.

Where ‘U’ denotes the content (tokens) of the input (formerly ‘x’),

And ‘P’ refers to the Positional Embedding signal.

How does this change attention?

$$\omega = QK^T = XW_q(XW_k)^T$$

$$\hat{\omega} = ((U + P)W_q)((U + P)W_k)^T$$

Expanding the Expression Yields four Terms

$$\hat{\omega} = ((U + P)W_q)((U + P)W_k)^T$$

$$\hat{\omega} = (UW_q)(UW_k)^T + (UW_q)(PW_k)^T + (PW_q)(UW_k)^T + (PW_q)(PW_k)^T$$

Attention Expansion w/ Added Signal ‘P’

$$\hat{\omega} = (UW_q)(UW_k)^T + (UW_q)(PW_k)^T + (PW_q)(UW_k)^T + (PW_q)(PW_k)^T$$

This creates 4 terms, which encode semantically different contributions.

1. The first represents Content-Content encoding (same as Self-Attention)
2. Content-Pos (Hybrid): Middle Terms 2 & 3
3. Pos-Pos: The final term (4), encodes purely positional relationships

Nearly all subsequent Positional Encoding Ideas emphasize one or more of these!!!!

Sinusoidal Encoding

Laying our PE Foundation

Sinusoidal Encoding (2017) - Definition

$$P_{[i,j]} = \begin{cases} \sin\left(10000^{\frac{-j}{D}} i\right), & \text{if } j \text{ is even} \\ \cos\left(10000^{\frac{-j}{D}} i\right), & \text{if } j \text{ is odd} \end{cases}, P \in \mathbb{R}^{N \times D}$$

“...each dimension of the positional encoding corresponds to a sinusoid…

***We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions**, since for any fixed offset k, Ppos+k can be represented as a linear function of Ppos.”*

- Vaswani et Al (2017).

Sinusoidal Encoding - Mathematical Intuition

Sinusoidal Encoding ‘P’ has the following key property: it can capture both ABSOLUTE AND RELATIVE positional information.

This is because:

1. Each position in the sequence is assigned a unique, fixed vector based on its absolute location relative to the start of the sequence.
2. By ***trigonometric properties*** P can express relative positional information.

Due to (1) P is considered to be an **absolute** encoding method.

Sinusoidal Encoding - Pos-Pos Term Expansion

$$(PW_q)(PW_k)^T = PW_qW_k^TP^T = P(W_qW_k^T)P^T = PWP^T$$

$$(PWP^T)_{[i,j]} = (P(WP^T))_{[i,j]} = \sum_{d=1}^{D} P_{[i,d]}(PW^T)_{[j,d]}$$

$$= \sum_{d=1}^{D} P_{[i,d]} \sum_{d'=1}^{D} P_{[j,d']}W^T_{[d,d']} = \sum_{d,d'=1}^{D} P_{[i,d]}P_{[j,d']}W^T_{[d,d']}$$

We end up with a weighted sum of the positional terms for the positional signal for tokens i and j.

Now applying this to the case of Sinusoidal Encoding ‘P’.

Sinusoidal Encoding - Achieving Relative Encoding Through Trigonometry

$$cos(\theta_1)cos(\theta_2) = \frac{1}{2}[cos(\theta_1 - \theta_2) + cos(\theta_1 + \theta_2)] \quad , sin(\theta_1)sin(\theta_2) = \frac{1}{2}[cos(\theta_1 - \theta_2) - cos(\theta_1 + \theta_2)]$$

$$cos(\theta_1)sin(\theta_2) = \frac{1}{2}[sin(\theta_1 + \theta_2) - sin(\theta_1 - \theta_2)] \quad , sin(\theta_1)cos(\theta_2) = \frac{1}{2}[sin(\theta_1 + \theta_2) + sin(\theta_1 - \theta_2)]$$

The above properties inform us that our **pos-pos products encode the relative distances between tokens through the differences in angles**.

This is because **the internal angles are functions of the token's position** and by the above properties their **products become functions which operate on relative differences and offsets**.

$$\theta_D(i,j) = 10000^{\frac{-j}{D}} i$$

Sinusoidal Encoding - Pos-Pos Term Expansion Example

Example Let $N = 2, D = 3, I = 2, J = 1$

$$
\begin{aligned}
(PWP^T)_{[2,1]} &= \sum_{d=1}^{D} P_{[i,d]} \sum_{d'=1}^{D} P_{[j,d']} W^T_{[d,d']} \\
&= \cos(\theta_D(2,1))[\cos(\theta_D(1,1))W^T_{[1,1} + \sin(\theta_D(1,2))W^T_{[1,2]} + \cos(\theta_D(1,3))W^T_{[1,3]}] \\
&+ \sin(\theta_D(2,2))[\cos(\theta_D(1,1))W^T_{[2,1]} + \sin(\theta_D(1,2))W^T_{[2,1]} + \cos(\theta_D(1,3))W^T_{[2,3]}] \\
&+ \cos(\theta_D(2,3))[\cos(\theta_D(1,1))W^T_{[3,1]} + \sin(\theta_D(1,2))W^T_{[3,2]} + \cos(\theta_D(1,3))W^T_{[3,3]}]
\end{aligned}
$$

As we can see each cell in the Pos-Pos tensor:

- Contain a (DxD) series of products, expressible as relative offsets in terms of the angle theta
- This ultimately encodes a linear of function of distance and offsets.
- All terms are scaled by learnable weights.

Sinusoidal Encoding - Pos-Content

$$(UW_q)(PW_k)^T = UW_qW_k^TP^T = U(W_qW_k^T)P^T = UWP^T$$

$$(UWP^T)_{[i,j]} = (U(WP^T))_{[i,j]} = \sum_{d=1}^{D} U_{[i,d]}(PW^T)_{[j,d]}$$

$$= \sum_{d=1}^{D} U_{[i,d]} \sum_{d'=1}^{D} P_{[j,d']}W^T_{[d,d']} = \sum_{d,d'=1}^{D} U_{[i,d]}P_{[j,d']}W^T_{[d,d']}$$

Looking at a cell in the first of our Pos-Content terms, we see that for a given cell (i-token, j-token), our added signal ‘P’ will result in a term which encodes the content of our token with the absolute position j.

The converse holds for the other Pos-Content term, which encodes the absolute position of our token i, with the content of our other token j.

Sinusoidal Encoding - High Level: ‘Why it Works’

At a high level introducing ‘P’ in this way encodes the following into our attention operation in a linear relationship:

- Two NxN Tensors which takes into account the content and absolute position of each other token pair in the input
- An NxN Tensor which takes into account the relative differences and offsets of different token positions.

…Which the learning process will then optimize.

Sinusoidal Encoding - Problems

We implicitly create a massive learnable expression in the attention operation which captures relative positional differences…

While we directly encode positional information, our process for encoding distance is noisy and includes a lot of contributions.

However… do these differences actually have desirable properties?

Sinusoidal Encoding - Problems

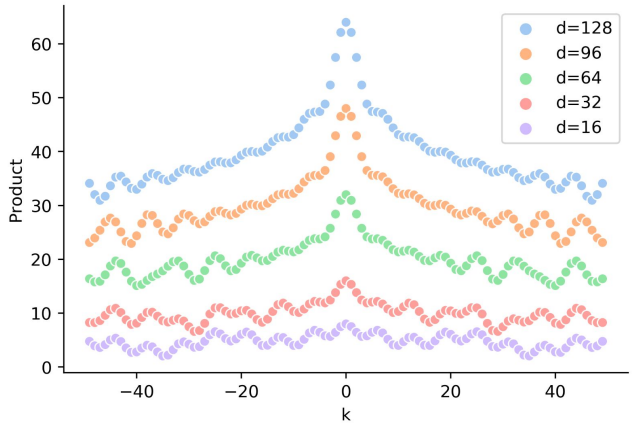


Figure 3: Dot product between two sinusoidal position embeddings whose distance is k. It is clear that the product is symmetrical, and with the increment of $|k|$, it has a trend to decrease, but this decrease is not monotonous.

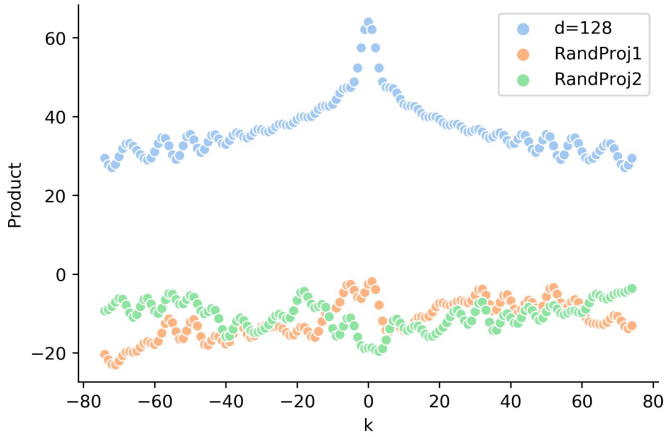


Figure 4: The upper line is the product between $PE_t^T PE_{t+k}$. The lower two lines are the products of $PE_t^T W PE_{t+k}$ with two random Ws. Although $PE_t^T PE_{t+k}$ can reflect the distance, the $PE_t^T W PE_{t+k}$ has no clear pattern.

Figures from: “TENER: Adapting Transformer Encoder for Named Entity Recognition”, Yan et Al, 2019

Sinusoidal Encoding - Problems

We easily ‘lose control’ of what happens to the positional signal in this setup, and leave a lot up to the learning process.

Absolute encodings on their own are frail and suffer from poor generalizability.

In this case, even the relative positional encodings will not clearly exhibit the desirable property of decay.

Sinusoidal Encoding - Some Good Questions!

What if our learned properties of distance disagree with prior knowledge of the domain?

Since we rely so much on the learning process in this setup, why not introduce dedicated learning parameters?
How few can we get away with without losing performance?

Do we really only care about distance and absolute position?
Why not directionality as well?

Is there a conceptual way of systematically approaching the four positional and content terms that are created from adding to an input?

Conceptual Trajectories in Positional Encoding

Absolute Positional Encoding

Absolute Positional Encoding

$$P \in \mathbb{R}^{N \times D}$$

Forms an NxD matrix of Learnable Parameters

Each position up to the MAX context length of the model receives a D-vector

This is simple to implement in practice, but introduces a LARGE number of parameters depending on the maximum context length and embedding size

Absolute Positional Encoding

$$P \in \mathbb{R}^{N \times D}$$

Compared alongside Sinusoidal Encoding, but with no clear performance gains

Most modern LLMs do not use Absolute Positional Encoding, and instead favor more motivated applications of learnable parameters as we shall see.

Absolute Positional Encoding

$$P \in \mathbb{R}^{N \times D}$$

Still used today in BERT models and certain other transformer architectures

Smaller, and Task-Specific models leverage Absolute Learnable Parameters

This is typically done in Encoder-Transformers where the maximum context length is fixed and smaller than in generative, Decoder-style models

Absolute Positional Encoding

$$P \in \mathbb{R}^{N \times D}$$

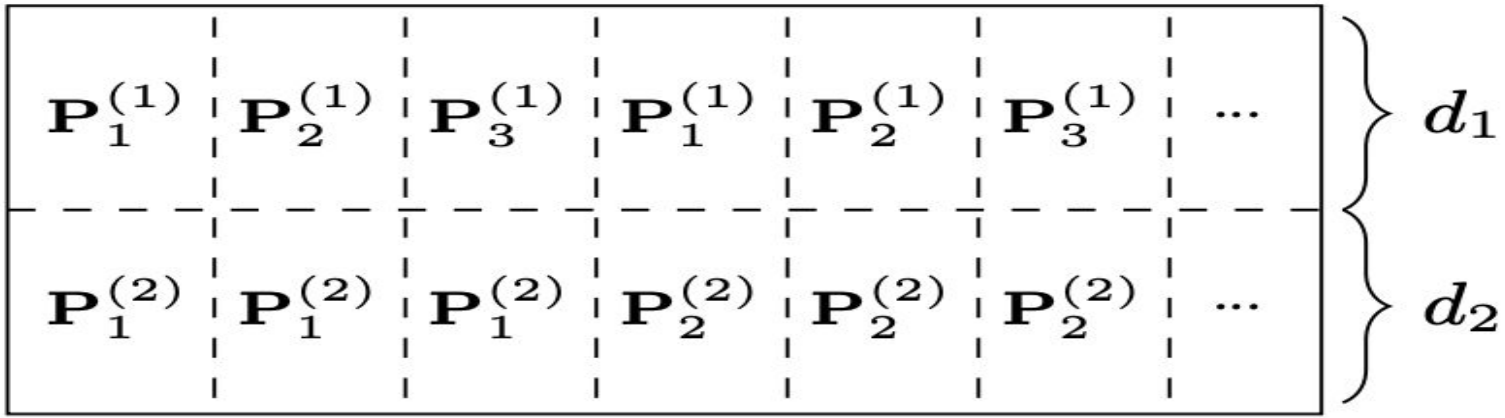
Early Vision transformers used Absolute Positional Encoding - some still do

Also appear in time-series models - sometimes at specific ranges which domain knowledge indicates can be volatile

Axial Encoding

Kitaev et Al (2020)

Axial Encoding (201X)

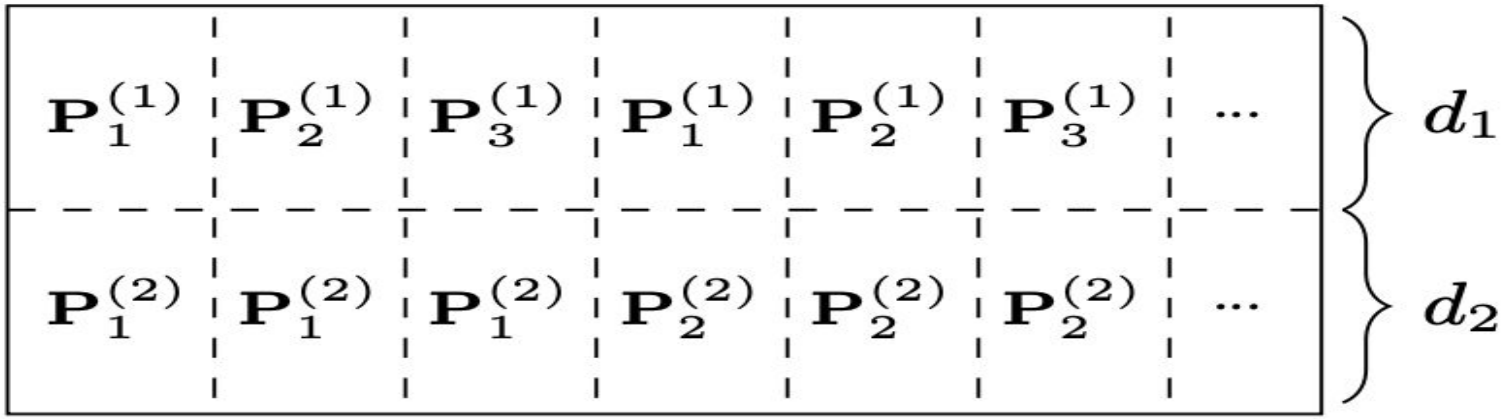


Applies learnable parameters such that different dimensions will attend to different groupings of elements.

Axial Encoding creates separate Positional Encodings for both individual elements (P1) AND segments (P2) based on absolute distance.

Figure from: “Position Information in Transformers: An Overview”, Dufter et Al, 2021

Axial Encoding (201X)



P1 captures the position within some predefined range, encoding segment-level positioning information (P1 has unique positions in some range repeating)

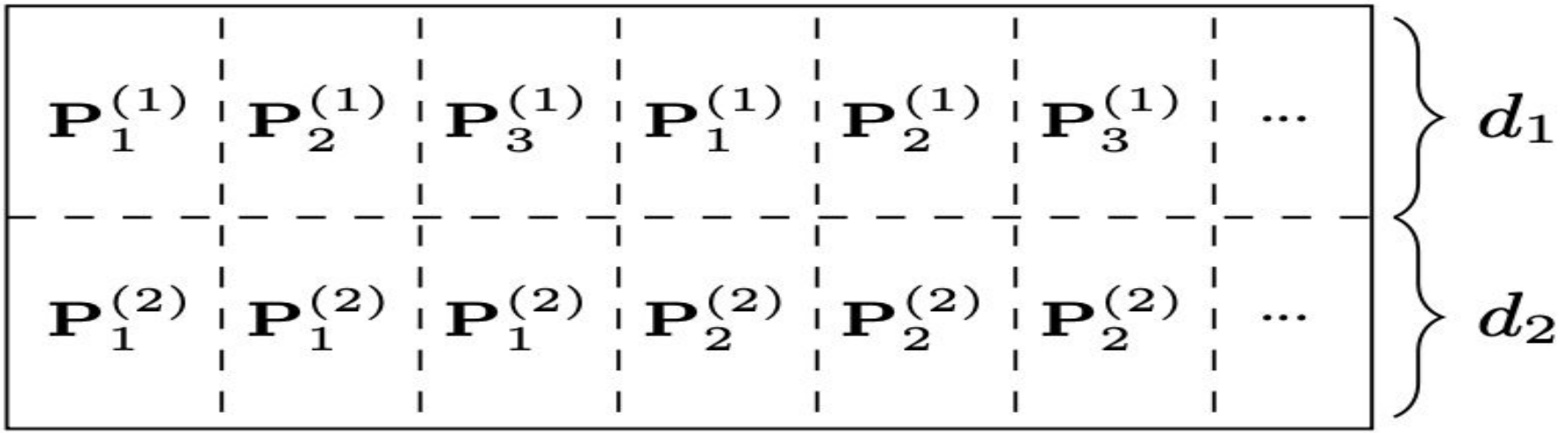
P2 encodes which segment the position is in separately

Figure from: “Position Information in Transformers: An Overview”, Dufter et Al, 2021

Figure from: “Position Information in Transformers: An Overview”, Dufter et Al, 2021

Axial Encoding Definition

$$P_{[n,d]} = \begin{cases} P_{1[r,d]}, & \text{if } d \leq d_1 \\ P_{2[s,d-d_1]}, & \text{if } d > d_1 \end{cases}$$



Axial Encoding Definition

$$P_{[n,d]} = \begin{cases} P_{1[r,d]}, & \text{if } d \leq d_1 \\ P_{2[s,d-d_1]}, & \text{if } d > d_1 \end{cases}$$

The way that both segment relative, AND segment identifying information is simultaneously encoded is via partitioning the embedding space in two.

All dimensions at or below some value d1 will encode segment-relative info

All dimensions above this value will encode segment-identifying info

Axial Encoding Definition

$$P_{[n,d]} = \begin{cases} P_{1[r,d]}, & \text{if } d \leq d_1 \\ P_{2[s,d-d_1]}, & \text{if } d > d_1 \end{cases}$$

Assumptions:

$$r = n \bmod n_1$$

$$s = \left\lfloor \frac{n}{n_1} \right\rfloor$$

Axial Encoding - Indexing Transformations

$$r = n \bmod n_1$$

$$s = \left\lfloor \frac{n}{n_1} \right\rfloor$$

The transformation ‘r’ makes sure that all token positions under P1 stay within the segment length n1.

This forms the range, with n1 acting as the maximum segment length

Axial Encoding - Indexing Transformations

$$r = n \bmod n_1$$

$$s = \left\lfloor \frac{n}{n_1} \right\rfloor$$

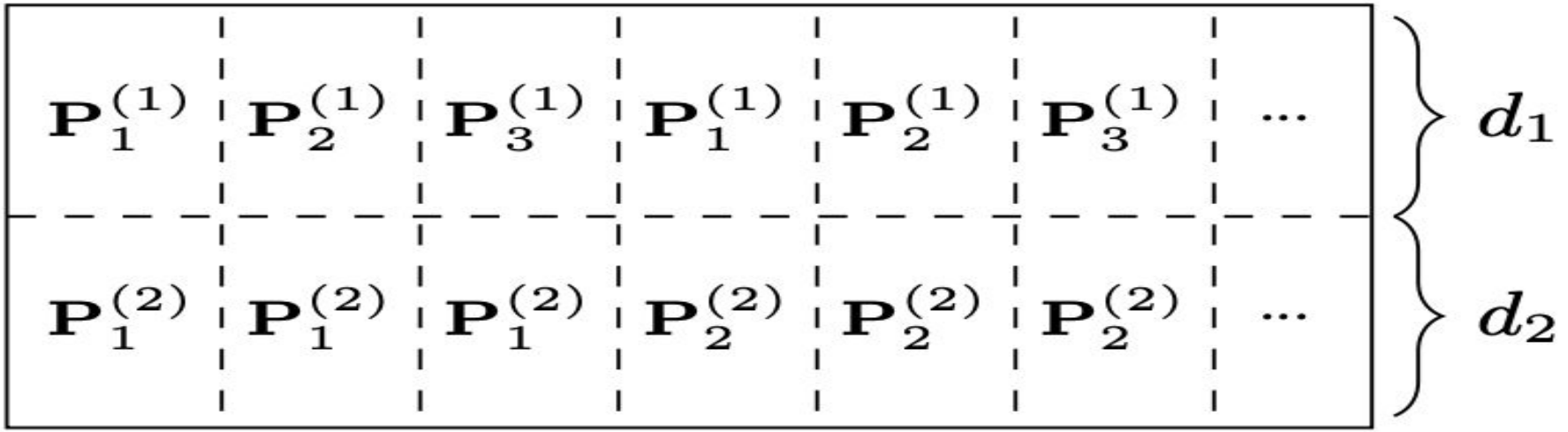
Transformation ‘s’ is what defines which segment the token is actually in.

Note: This defines that all segments are of the same length n1.

Figure from: “Position Information in Transformers: An Overview”, Dufter et Al, 2021

Axial Encoding Example: (Let n1 = 3, d1=d2=1, D=2)

$$P_{[n,d]} = \begin{cases} P_{1[r,d]}, & \text{if } d \leq d_1 \\ P_{2[s,d-d_1]}, & \text{if } d > d_1 \end{cases}$$



Underpinning Axial Encoding are two key assumptions

$$\text{Let } N_{max} = n_1 n_2, \text{ and } D = d_1 + d_2,$$

$$\text{such that: } P_1 \in \mathbb{R}^{n_1 \times d_1} \text{ and } P_2 \in \mathbb{R}^{n_2 \times d_2}$$

Axial Encoding - Ensuring Max Context Length

$$\text{Let } N_{max} = n_1 n_2, \text{ and } D = d_1 + d_2,$$

$$\text{such that: } P_1 \in \mathbb{R}^{n_1 \times d_1} \text{ and } P_2 \in \mathbb{R}^{n_2 \times d_2}$$

The first is trivial:

That the maximum context length needs to be divisible by the segment length.

This defines n2 total segments.

Axial Encoding - Ensuring Max Context Length

$$\text{Let } N_{max} = n_1 n_2, \text{ and } D = d_1 + d_2,$$

$$\text{such that: } P_1 \in \mathbb{R}^{n_1 \times d_1} \text{ and } P_2 \in \mathbb{R}^{n_2 \times d_2}$$

The second serves two roles:

As stated before - it allows us to encode two positional signals into the same token

It also assists in reducing the total number of parameters defined…

Axial Encoding - Fewer Parameters

$$\text{Let } N_{max} = n_1 n_2, \text{ and } D = d_1 + d_2,$$

$$\text{such that: } P_1 \in \mathbb{R}^{n_1 \times d_1} \text{ and } P_2 \in \mathbb{R}^{n_2 \times d_2}$$

Combined with our definition of ‘P’, we can define fewer than N x D parameters.

Specifically we define: n1xd1 + n2xd2

Axial Encoding - Content-Pos

$$(UW_q)(PW_k)^T = UW_qW_k^TP^T = U(W_qW_k^T)P^T = UWP^T$$

$$(UWP^T)_{[i,j]} = (U(WP^T))_{[i,j]} = \sum_{d=1}^{D} U_{[i,d]}(PW^T)_{[j,d]}$$

$$= \sum_{d=1}^{D} U_{[i,d]} \sum_{d'=1}^{D} P_{[j,d']}W^T_{[d,d']} = \sum_{d,d'=1}^{D} U_{[i,d]}P_{[j,d']}W^T_{[d,d']}$$

Content-Pos Term Expansion

The above product informs us that both the segment-wise AND segment position tensors for the target position are encoded with the query's content.

The reverse is also encoded through the Pos-Content.

Axial Encoding - Pos-Pos

$$(PWP^T)_{[i,j]} = (P(WP^T))_{[i,j]} = \sum_{d=1}^{D} P_{[i,d]}(PW^T)_{[j,d]}$$

$$= \sum_{d=1}^{D} P_{[i,d]} \sum_{d'=1}^{D} P_{[j,d']} W^T_{[d,d']} = \sum_{d,d'=1}^{D} P_{[i,d]} P_{[j,d']} W^T_{[d,d']}$$

Pos-Pos Term Expansion

The above product informs us that both the segment-wise AND segment position tensors are cross encoded with each other for the query and key positions.

Axial Encoding - Problems

- Unlike Sinusoidal encoding, these terms do not directly encode absolute positional information into the attention expression.

- The ability to account for relative differences is questionable (indirect at best).
  - We only encode ‘relative to the segment’s starting point’ and the segment id’s.

- All segments are forced to be the same length - n1

- Still creating a significant amount of parameters for large N and D

Overview of Problems with APE - In General

Potency: Signal loses direct influence as depth increases (added at start only).
Note: In some VERY SPECIFIC cases, this could be a desirable feature

Size Intensive: do we really need NxD items?.

Unintentional Complications:

Pos-Pos can be affected by a cacophony of influences on it’s own.

Content-AbsolutePos may not even be the very desirable to encode!

Positional Encoding: Early Years

2018-2021

Core questions in this Era:

Are absolute sinusoids from Vaswani et al. enough?

How do we encode relative distance so the model generalizes to new lengths?

Should position enter as extra embeddings (added to tokens) or as biases on attention?

How do we adapt PEs to longer contexts (Transformer-XL), and to new modalities (vision, speech)?

High-Level Map (Focusing on NLP)

‘Era’ starts with the work of Peter Shaw et Al (2018) and the RPR Method

Other methods of this time largely fall within the tradition of this work

Focused on what kind of positional information to inject (absolute vs relative, embeddings vs biases) and how to make that efficient for language, translation, and early vision models.

Lots of tinkering: absolute vs relative, embeddings vs biases…

The big “end-of-era” move is ALiBi and RoPE, showing that positional design is central to extrapolation and can be extremely lightweight.

Move Towards Attention-Based Encodings

Move Towards Attention-Based Encodings

Various aforementioned limitations inspired a move towards introducing the Positional Signal directly into the attention operation itself… Which:

- Greatly reduced the sizes of the tensors involved.
- Improves control over positional signal (now manual).
- Eliminates the need for influence by unwanted noise/terms.
- Greatly simplifies analysis.
- Adding positional information at each layer improves performance.

(Research has generally not looked back since.)

Self-Attention with Relative Position Representations

Shaw et al (2018)

Emphasizing Relative Distance in Positional Encoding

The major focus of Shaw et al's work is the emphasis on relative distance between elements as opposed to absolute position information.

Shaw et al found that **combining relative and absolute information yielded no significant gains** over **solely relative position information**.

This finding would go on to inspire a large family of positional encoding methods which generally prioritize the use of relative distances in their formulation.

Intuitive View - Nodes and Edges

$$\text{Let } a_{[i,j]} = w_{j-i}, \text{ and } w = (w_{1-k}, .., w_0, ...w_{k-1}) \text{ and } w_x \in \mathbb{R}^D$$

Consider each element as nodes, and the edges between them representing their relative distances from each other.

The goal is to learn the optimal values these vectors in the embedding space, with each relative distance (up to some max distance - k).

Positional encoding signals are thus learned and shared on the basis of relative distance from one another.

Cell-Level View: What's new?

$$\text{Let } a_{[i,j]} = w_{j-i}, \text{ and } w = (w_{1-k}, .., w_0, ...w_{k-1}) \text{ and } w_x \in \mathbb{R}^D$$

$$\hat{\omega}_{[i,j]} = Q_{[i]} \cdot K_{[j]} + Q_{[i]}(a_{[i,j]})^T$$

We have an NxNxD tensor, a, whose cells map to a set of learnable vectors.

Our new score contains two terms, our dot-product (as before) and a bias term

This bias term constitutes the intervention introduced by Shaw et Al.

Cell-Level View: Dot Product Notes

$$\text{Let } a_{[i,j]} = w_{j-i}, \text{ and } w = (w_{1-k}, .., w_0, ...w_{k-1}) \text{ and } w_x \in \mathbb{R}^D$$

$$\hat{\omega}_{[i,j]} = Q_{[i]} \cdot K_{[j]} + Q_{[i]} a^I_{[i,j]} = Q_{[i]} \cdot K_{[j]} + Q_{[i]} \cdot a_{[i,j]}$$

Shaw's bias essentially creates two dot-products in each i,j cell

One is standard to self-attention, as we went over before…

This new one is between the query-token (i) and the learned distance representation for the relative distance between Qi and it's target position (j).

Essentially *we are learning our Query Row (i)'s representation with the relative distance between it's lookup target (j) - and this is applied as a bias to the score*

Tensor-Level View: Attention Reformulation

$$\hat{\omega} = (UW_q)(UW_k)^T + (UW_q)(PW_k)^T + (PW_q)(UW_k)^T + (PW_q)(PW_k)^T$$

$$\hat{\omega} = QK^T + einsum(Q_{id}, a_{ijd}; \rightarrow_{ij})$$

At a higher level, what we have essentially rewritten our expanded formulation of attention in much simpler form by trading three bias terms, for one!

Tensor-Level View: Attention Reformulation

$$\hat{\omega} = (UW_q)(UW_k)^T + (UW_q)(PW_k)^T + (PW_q)(UW_k)^T + (PW_q)(PW_k)^T$$

$$\hat{\omega} = QK^T + einsum(Q_{id}, a_{ijd}; \rightarrow_{ij})$$

Our new term explicitly contains relative position information by mapping a vector to each relative distance pair - replacing the Pos-Pos term in sinusoidal encoding

Furthermore, our term also encodes content-position relations via the Q tensor.

Shaw’s Experiment

Motivation:

Determine whether relative positional representations improve translation vs. sinusoidal absolute encodings.

Task:

Supervised neural machine translation (NMT) with Transformer encoder–decoder architectures.

Metric: BLEU

Shaw's Experiment

Model	Position Information	EN-DE BLEU	EN-FR BLEU
Transformer (base)	Absolute Position Representations	26.5	38.2
Transformer (base)	Relative Position Representations	**26.8**	**38.7**
Transformer (big)	Absolute Position Representations	27.9	41.2
Transformer (big)	Relative Position Representations	**29.2**	**41.5**

Table 1: Experimental results for WMT 2014 English-to-German (EN-DE) and English-to-French (EN-FR) translation tasks, using newstest2014 test set.

Transformative Reformulation!

Shaw's specific reformulation of attention would inspire a variety of approaches.

In particular, we will often see:

1. Learnable relative position mappings
    - Learnable elements of desirable sizes (scalar, vector, etc.)
2. Distance clipping
3. Formulation of custom content-relativePosition bias terms
4. A relative distance oriented philosophy

Not only WHAT to add, but WHERE as well!

Shaw also applies a similar expression to the context tensor Z via the V tensor.

This term introduced no measurable effect… and future methods which improve on Shaw's design do not adopt this.

It was a significant finding, that this term largely irrelevant, and the major performance gains were found in the attention score computation step

Notes - What was left out and why

To avoid a direct computation on an NxNxD tensor, Shaw et Al employ an efficient lookup method to reduce the size of the needed tensors.

We employ a direct formulation for conceptual clarity.

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Dai et Al (2019)

Intuition + Research Question

Transformers are often limited by their context length

Is there a way to model dependencies beyond a fixed context length?

Transformer-XL (2019): Intro

Expands on Shaw's relative encoding scheme

Introduced a few novel concepts:

- Having separate parameters for Content and Positional signal terms.
- Recreating all four terms of the expanded attention expression to capture global biases
- Feeding segment-level recurrences into the content tensors (we leave this out for simplicity)

Cell-Level View: What's different from Shaw?

$$\text{Let } \hat{R} \in \mathbb{R}^{N \times N \times D} \text{ s.t. } \hat{R}_{[i,j]} = R_{[i-j]} W_k^{(P)} \text{ where } R \in \mathbb{R}^{N \times D}$$

$$\text{Let } u, v \in \mathbb{R}^{D}$$

$$\hat{\omega}_{[i,j]} = Q_{[i]} \cdot K_{[j]} + \sum_{d}^{D} Q_{[i,d]} \hat{R}_{[i,j,d]} + \sum_{d}^{D} u_{[d]} K_{[j,d]} + \sum_{d}^{D} v_{[d]} \hat{R}_{[i,j,d]}$$

Separate set of weights dedicated to learning the positional relationships.

This enriches the linear expression of each relative distance vector.

Cell-Level View: What's going on here?

$$\text{Let } \hat{R} \in \mathbb{R}^{N \times N \times D} \text{ s.t. } \hat{R}_{[i,j]} = R_{[i-j]} W_k^{(P)} \text{ where } R \in \mathbb{R}^{N \times D}$$

$$\text{Let } u, v \in \mathbb{R}^{D}$$

$$\hat{\omega}_{[i,j]} = Q_{[i]} \cdot K_{[j]} + \sum_{d}^{D} Q_{[i,d]} \hat{R}_{[i,j,d]} + \sum_{d}^{D} u_{[d]} K_{[j,d]} + \sum_{d}^{D} v_{[d]} \hat{R}_{[i,j,d]}$$

We have two additional bias terms, each containing a learnable vector, and each attends to either the column token or the relative distance respectively.

Tensor-Level View

$$\hat{\omega} = QK^T + einsum(Q_{id}, \hat{R}_{ijd}; \rightarrow_{ij}) + einsum(u_d, K_{jd}; \rightarrow_j) + einsum(v_d, \hat{R}_{ijd}; \rightarrow_{ij})$$

Setting aside the segment-level recurrences, the major changes can be expressed simply using these two additional bias terms

At a high level these bias terms are intended to, respectively, account for:

1. Global content bias
2. Global positional bias

Note: These terms are ‘global’ in that they are query-agnostic

Bias Breakdown: Why Global Content?

$$\hat{\omega} = QK^T + einsum(Q_{id}, \hat{R}_{ijd}; \rightarrow_{ij}) + einsum(u_d, K_{jd}; \rightarrow_j) + einsum(v_d, \hat{R}_{ijd}; \rightarrow_{ij})$$

The reasoning here is that a bias attending to different words should not be based on the position of the query term.

Note: By the dimensionality of the this term, it is broadcast across all query rows

Bias Breakdown: Why Global Positional?

$$\hat{\omega} = QK^T + einsum(Q_{id}, \hat{R}_{ijd}; \rightarrow_{ij}) + einsum(u_d, K_{jd}; \rightarrow_j) + einsum(v_d, \hat{R}_{ijd}; \rightarrow_{ij})$$

A similar term replaces the Pos-Pos term, with a learnable vector that simply maps to our relative distance:

The old purely positional terms is now solely dependent on a learned relative distance relationship between tokens.

Bias Coverage: Global vs Local & Content vs Position

$$\hat{\omega}_{[i,j]} = Q_{[i]} \cdot K_{[j]} + \sum_{d}^{D} Q_{[i,d]} \hat{R}_{[i,j,d]} + \sum_{d}^{D} u_{[d]} K_{[j,d]} + \sum_{d}^{D} v_{[d]} \hat{R}_{[i,j,d]}$$

There is conceptual coverage in differentiating between ‘global’ and ‘local’ via the inclusion of positional information from token ‘i’ in our terms.

Viewed from this perspective, our expression now accounts for all four combinations of local vs global, and content vs (relative) positional biases.

Transformer-XL Experiment: Motivations

Shaw’s relative position attention cannot handle context beyond a fixed segment length (due to clipped distances and no recurrence).

RPR is incompatible with caching, preventing efficient reuse of hidden states.

Transformer-XL aims to extend relative attention to long-range dependencies by introducing a recurrence mechanism and a reformulated positional term.

Transformer-XL Experiment: Tasks + Metrics

Tasks:

Language Modeling, specifically: WikiText-103 (word-level LM)

Shaw's model is inserted into Transformer-XL's ablations as a baseline attention mechanism to compare against.

Metrics:

Perplexity (PPL) – for WikiText-103

Transformer-XL outperforms Shaw-style relative positional attention.

Table from: “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”, Dai et Al, 2019

Transformer-XL Experiment: Results

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	**27.02**	**26.77**	**500**
-	✓	Shaw et al. (2018)	Full	27.94	27.94	256
-	✓	Ours	Half	28.69	28.33	460
-	✗	Ours	Full	29.59	29.02	260
-	✗	Ours	Half	30.10	30.10	120
-	✗	Shaw et al. (2018)	Full	29.75	29.75	120
-	✗	Shaw et al. (2018)	Half	30.50	30.50	120
-	✗	Vaswani et al. (2017)	Half	30.97	30.97	120
Transformer (128M)†	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
Transformer-XL (151M)	✓	Ours	Full	23.43	**23.09**	**640**
					23.16	450
					23.35	300

Table 6: Ablation study on WikiText-103. For the first two blocks, we use a slightly smaller model (128M parameters). † indicates that the corresponding row is reduced to the same setting as the Transformer network in (Al-Rfou et al., 2018), except that two auxiliary losses are not implemented in our experiments. “PPL init” refers to using the same length as training. “PPL best” indicates the perplexity obtained by using the optimal length. “Attn Len” is the shortest possible attention length during evaluation to achieve the corresponding result (PPL best). Increasing the attention length during evaluation improves performance only when our positional encoding is used. The “Transformer-XL (151M)” setting uses a standard parameter budget as previous work (Merity et al., 2018), where we observe a similar effect when increasing the attention length during evaluation.

Table from: “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”, Dai et Al, 2019

Transformer-XL Experiment: Results

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	**27.02**	**26.77**	**500**
-	✓	Shaw et al. (2018)	Full	27.94	27.94	256
-	✓	Ours	Half	28.69	28.33	460
-	✗	Ours	Full	29.59	29.02	260
-	✗	Ours	Half	30.10	30.10	120
-	✗	Shaw et al. (2018)	Full	29.75	29.75	120
-	✗	Shaw et al. (2018)	Half	30.50	30.50	120
-	✗	Vaswani et al. (2017)	Half	30.97	30.97	120
Transformer (128M)†	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
Transformer-XL (151M)	✓	Ours	Full	23.43	**23.09**	**640**
					23.16	450
					23.35	300

Transformer-XL’s formulation outperforms RPR both with and without recurrence

However, as we can see, the use of segment-level recurrences seems to have a much stronger effect than the two new bias terms alone

Table from: “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”, Dai et Al, 2019

Ablation Caveat! Recurrence does a lot of lifting

Method	PPL
Ours	**25.2**
With Shaw et al. (2018) encodings	25.7
Without recurrence	27.1

Table 7: Ablation study on One Billion Word, a dataset without long-term dependency.

This effect is exaggerated on datasets without long-term dependencies

Here the global biases seem to do more harm than good without recurrence

Transformer-XL (2019): Comments/Concerns

Segment-level recurrences has not been widely adopted in purely ‘PE’ contexts.

Primarily a means for navigating the specific context-length limitation problem.

Why have a seperate DxD matrix of weights when they are only applied to the positional vectors - which are already learnable and of shape D? (It’s all linear)

TENER: Adapting Transformer Encoder for Named Entity Recognition

Yan et Al (2019)

TENER Encoding (Distance + Direction + Fixed Signal)

Conceptually expands on both Shaw and Transformer-XL (w/o recurrence)

Resurfaces sinusoidal waves, uses the concept as a means of explicitly encoding both distance and directionality into the attention biases

Combines:

1. Transformers-XL's expanded bias terms for global content and position bias
2. A fixed distance bias with learnable parameters

TENER's performance greatly surpasses the Vanilla Transformer in NER tasks.

Unfortunately, the paper does not compare against Shaw et Al or Transformer-XL.

Return of Sinusoidal Waves

Where $u, k \in \mathbb{R}^{D_k}$, and $P \in \mathbb{R}^{N \times D_k}$ is defined such that:

$$P_x = \begin{bmatrix} \sin(c_0 x) \\ \cos(c_0 x) \\ \sin(c_1 x) \\ \cos(c_1 x) \\ \vdots \\ \sin(c_{(\frac{D_k}{2}-1)} x) \\ \cos(c_{(\frac{D_k}{2}-1)} x) \end{bmatrix}, P_{-x} = \begin{bmatrix} -\sin(c_0 x) \\ \cos(c_0 x) \\ -\sin(c_1 x) \\ \cos(c_1 x) \\ \vdots \\ -\sin(c_{(\frac{D_k}{2}-1)} x) \\ \cos(c_{(\frac{D_k}{2}-1)} x) \end{bmatrix}$$

TENER Cell Level View: Departures from Shaw & XL

$$\text{Let } \hat{P} \in \mathbb{R}^{N \times N \times D} \text{ s.t. } \hat{P}_{[i,j,d]} = P_{[i-j,d]}$$

$$\hat{\omega}_{[i,j]} = \sum_{d}^{D} Q_{[i,d]} K_{[j,d]} + Q_{[i,d]} \hat{P}_{[i,j,d]} + u_{[d]} K_{[j,d]} + v_{[h,d]} \hat{P}_{[i,j,d]}$$

Here the position tensor is NOT learnable, and instead encodes a fixed signal P.

Enforces a non-monotonic decay of values as the distance between i & j grows.

P’s values will also change depending on the directionality of the tokens as well.

TENER Tensor Level View:

$$\hat{\omega} = QK^T + einsum(Q_{id}, \hat{P}_{ijd}; \rightarrow_{ij}) + uK^T + einsum(v_d, \hat{P}_{ijd}; \rightarrow_{ij})$$

Shaw and XL's learned position-content bias is altered to include a fixed position tensor: this reduces parameters but also enforces the desired signal properties.

Global content bias remains unchanged from Transformer-XL.

XL's global position bias also adopts a fixed position tensor.

TENER - Takeaways

Sinusoidal waves can elegantly capture directionality while enforcing decay

Reminder: Directionality can also be encoded into the embedding space

Global bias terms appear effective when applied through a fixed signal in the embedding space, at least given the results from Wu et Al (2021)

Hybrid expressions of fixed and learnable attention bias seem comparable in performance into parameter intensive methods

Improve Transformer Models with Better Relative Position Embeddings

Huang et al (2020)

Huang et al (2020)

Primarily Expands on Shaw's relative encoding.

(Abandons XL's formulation without segmented recurrence)

Defines four methods which are conceptually similar.

Compares all four methods, alongside Shaw, and compares in several tasks.

Similar to an ablation of conditions inspired by Shaw's formulation.

Huang - Methods 1 & 2

$$\hat{\omega}_{[i,j]} = (QK^T)_{[i,j]} a_{[i,j]}, \text{ where } a_{[i,j]} = w_{|j-i|}, w = (w_0, \ldots w_{k-1}) \text{ and } w_x \in \mathbb{R}.$$

$$\hat{\omega}_{[i,j]} = (QK^T)_{[i,j]} a_{[i,j]}, \text{ where } a_{[i,j]} = w_{j-i}, w = (w_{1-k}, .., w_0, \ldots w_{k-1}) \text{ and } w_x \in \mathbb{R}.$$

Here each attention cell (i,j) is directly scaled by a learnable scalar which corresponds to the relative distance between token's i and j.

Method 1 limits this to absolute distance.

Method 2 allows for directionality to be considered.

Huang - Methods 1 & 2

$$\text{Let } a \in \mathbb{R}^{N \times N}$$

$$\hat{\omega} = (QK^T) \odot a$$

This is easily formulable at a high level using a relative position matrix.

Here the attention scores depart from convention as rather than an additional bias term, the scores themselves are directly scaled individually

Each learned scalar corresponds to each token’s pairwise relative distance.

Huang - Method 3

$$\text{Let } a \in \mathbb{R}^{N \times N \times D}$$

$$\text{Let } a_{[i,j]} = w_{j-i},\ w = (w_{1-k}, .., w_0, ...w_{k-1}) \text{ and } w_x \in \mathbb{R}^D$$

$$\hat{\omega}_{[i,j]} = \sum_{d=1}^{D} Q_{[i,d]} K_{[j,d]} a_{[i,j,d]}$$

$$\hat{\omega} = einsum(Q_{id}, K_{jd}, a_{ijd}; \rightarrow_{ij})$$

Similar to methods 1&2, except the new learnable term is now a vector

Each vector is introduced directly into the normal dot-product attention of each cell

Thus, each contribution from the Q and K tensor's embedding is scaled by a learnable scalar across the entire embedding space on the basis of distance

Huang - Method 4

$$\text{Let } a \in \mathbb{R}^{N \times N \times D}$$

$$\text{Let } a_{[i,j]} = w_{j-i},\ w = (w_{1-k}, .., w_0, ...w_{k-1}) \text{ and } w_x \in \mathbb{R}^D$$

$$\hat{\omega}_{[i,j]} = Q_{[i]} \cdot K_{[j]} + Q_{[i]} \cdot a_{[i,j]} + K_{[j]} \cdot a_{[i,j]}$$

$$\hat{\omega} = QK^T + einsum(Q_{id}, a_{ijd}; \rightarrow_{ij}) + einsum(K_{jd}, a_{ijd}; \rightarrow_{ij})$$

Most similar to Shaw's formulation, but with one addition

The content–position bias is replicated for the key tensor.

Experimentation: Motivation

The primary research question we are concerned with is:

Do the proposed relative position embeddings improve over Shaw et al. (2018) and absolute embeddings?

Huang et al claim Shaw's relative embedding is helpful, but still under-utilizes interactions between query, key, and relative distance.

They propose four new variants to increase this interaction.

Experimentation: Tasks

Huang et Al provide two primary tasks:

SQuAD v1.1 (Stanford Question Answering Dataset)

GLUE Benchmark (General Language Understanding Evaluation):

- MNLI (matched/mismatched) → natural language inference
- QQP → paraphrase identification (F1)
- SST-2 → sentiment classification
- MRPC → paraphrase classification (F1)

Experimentation: Tasks

SQuAD:

- EM (Exact Match)
- F1 score

GLUE tasks:

- Accuracy: MNLI, SST-2
- F1: QQP, MRPC

Table from: “Improve Transformer Models with Better Relative Position Embeddings”, Huang et Al, 2020

Huang - Performance Comparison SQuAD

Model	EM	F1
BERT (Devlin et al., 2018)	80.8	88.5
Absolute	81.58	88.59
(Shaw et al., 2018)	82.38	89.37
Method 1	80.82	87.96
Method 2	81.44	88.86
Method 3	**83.71**	90.50
Method 4	83.63	**90.53**

Table 2: SQuAD1.1 development results for various position embeddings on BERT-base.

Methods 1 and 2 generally underperformed relative to Shaw, but not absolute

Methods 3 and 4 were the highest performing methods overall

Method 4 was the most successful method in almost all experiments

Table from: “Improve Transformer Models with Better Relative Position Embeddings”, Huang et Al, 2020

Huang - Performance Comparison GLUE

Model	MNLI-(m/mm)	QQP	SST-2	MRPC
BERT (Devlin et al., 2018)	84.6/83.4	71.2	**93.5**	**88.9**
Absolute	83.57/83.65	87.64	90.48	88.40
(Shaw et al., 2018)	84.10/84.07	**87.77**	90.94	88.68
Method 1	83.84/84.06	87.52	91.97	88.65
Method 2	83.68/83.78	87.50	91.05	87.34
Method 3	**84.81/84.68**	87.11	91.39	82.86
Method 4	84.45/84.51	87.41	91.74	88.88

Table 3: GLUE development results for different position embeddings on BERT base setting.

Performance showed little variation across tasks

However, baseline methods often won out for these tasks

Huang - Takeaways

Scaling the attention scores directly by scalars is likely damaging to performance

Symmetrical content-position biases (M4) seem to improve performance over just a query-position bias term (Shaw)

Expanded sum-product (M3) formulation is competitive with two bias terms (M4)

T5 Bias

Raffel et Al (2020)

T5 Positional Bias

Adopts a novel formulation of positional encoding

Learns a single, scalar bias for each relative distance within a set of ranges

Authors define a logarithmic progression of 32 steps up to a max distance of 128.

Each of these steps creates a range, which corresponds to a learnable scalar

T5 Positional Bias

Intuition:

This creates ‘sensitive’ ranges around close terms to learn different dependencies, but as terms become more distant, they fall into similar relationships.

Note:

Bias terms are SHARED across layers

Each head learns its own set of biases

Cell Level View

$$\hat{\omega}_{[h,i,j]} = QK^{T}_{[h,i,j]} + B_{h,i-j}$$

This is a purely positional formulation of encoding

Each head has its own, learnable, scalar bias added

Each scalar is based on the relative distance between tokens and the range it falls under, according to the pre-defined progression

Tensor Level View

$$\hat{B} \in \mathbb{R}^{H \times N \times N}$$

$$\hat{\omega} = QK^T + \hat{B}$$

A direct formulation can simply define a tensor in HxNxN space

B here acts similar to a relative distance matrix

B maps to the scalar associated with the RANGE that distance falls under

T5 Notes

Introduces a simple and elegant formulation, as well as many novel concepts:

It was an early adopter of head-specific signals

Defining terms according to a progression of ranges, rather than specific distances

Reusing parameters across layers

T5 was a landmark paper, but is mostly relevant for it’s broad architectural experimentations rather than positional encoding innovations

It’s novelty has still been influential none-the-less

DA-Transformer: Distance-aware Transformer

Wu et Al (2021)

DA Transformer

Once again emphasizes relative distances between tokens

Expands on the scaling approach used in Huang M1/M2

Uses a carefully chosen sigmoid function instead of a scalar

Learns a different scaling factor per head (similar to T5)

Uses an activation to control self-attention scores prior to scaling

Compares Shaw et Al (RPR), Transformer-XL, TENER (Adapted) and DA

A Good Question: What is in an Ideal PE Function?

Our function accepts a relative distance |i-j| scaled by a learnable parameter w

1. f(0) = 1
   - Zero distances does not influence attention scores
2. f(x) = 0 as x approaches negative infinity
   - If the function prefers local information (w<0), distant info can be surpassed
3. f(x) should be limited as x approaches infinity
   - Able to process long distances without overemphasizing them
4. f(.) should have a tunable scale
   - The model should be able to adjust the intensity of distance information
5. f(.) needs to be monotonic

Custom Sigmoid Function to the Rescue!

$R \in \mathbb{R}^{N \times N}$ is a relative position matrix, where $R_{[i,j]} = |i - j|$

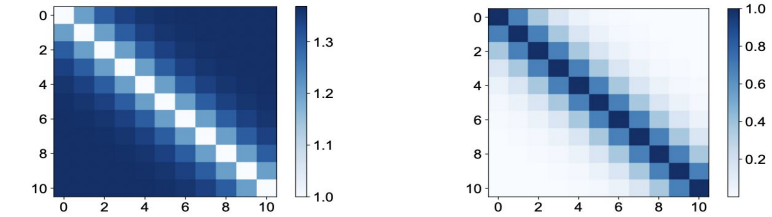
$$\hat{R} = \frac{1 + e^{v}}{1 + e^{v - (w \odot R)}}$$

$$v, w \in \mathbb{R}^{H}$$

1. When distance = 0, function is 1
2. As wR -> -inf, function becomes 0|
3. Function is upper bounded
4. The scale of the function is tuned by v
5. Sigmoids are monotonic

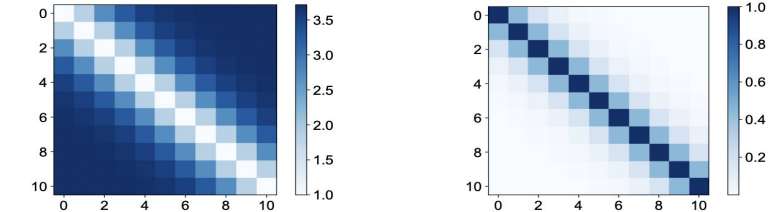
Figure from: “DA-Transformer: Distance-aware Transformer”, Wu et Al, 2021

Other Desirable Properties of the Sigmoid



(a) $w_i = 1, v_i = -1.$

(b) $w_i = -1, v_i = -1.$



(c) $w_i = 1, v_i = 1.$

(d) $w_i = -1, v_i = 1.$

Figure 2: The re-scaled coefficient matrices under different values of w_i and v_i. Dark regions indicate that the corresponding attention weights are promoted.

Function limits the reliance upon and total number of parameters

The learnable scalars v,w and relative distance values define a sigmoid

The different values of v and w emphasize the distances and ranges which are emphasized in the attention scores

Head Specific Sigmoid Functions

$v, w \in \mathbb{R}^H$

$R \in \mathbb{R}^{N \times N}$ is a relative position matrix, where $R_{[i,j]} = |i - j|$

$$\hat{R} = \frac{1 + e^{v}}{1 + e^{v - (w \odot R)}}, \hat{R_{[h]}} = \frac{1 + e^{v_{[h]}}}{1 + e^{v_{[h]} - (w_{[h]} \odot R)}}$$

DA Encoding is split across all the heads of the attention mechanism

This enables each head to learn to emphasize different ranges of dependencies

Some heads may learn distance & narrow, others close & wide, etc.

Cell Level View

Let $\phi = \text{RELU}$

$$\hat{\omega}_{[i,j]} = (\phi(QK^T) \odot \hat{R})_{[i,j]} = \text{RELU}(QK^T)_{[i,j]} \hat{R}_{[i,j]}$$

$$= \begin{cases} (QK^T)_{[i,j]} \hat{R}_{[i,j]}, & \text{if } (QK^T)_{[i,j]} > 0 \\ 0, & \text{if } (QK^T)_{[i,j]} \leq 0 \end{cases}$$

Each cell is scaled by the sigmoid value or zero

RELU cancels out all non-positive attention values

Tensor Level View

$$\hat{\omega} = \phi(QK^T) \odot \hat{R}$$

Why scale rather than add?

Prevents strong positional coefficients from overemphasizing weak attention cells.

However, scaling emphasizes both positive and negative values… which requires additional treatment.

This motivates the use of an activation function, namely RELU.

$$\text{Let } \hat{R} \in \mathbb{R}^{H \times N \times N}$$

RELU is used for the following reasons:

- Prevents position from scaling negatives:
  - Negative attention cell values will not be overemphasized
- Introduces sparsity
  - Spatial relations will only be learned between relevant tokens
- Possibly simplifies learning procedure for sigmoid function
  - Sigmoid only needs to learn relevant relationships

Performance Comparison

DA-Transformer outperforms previous major methods mentioned here:

1. Transformer (Vaswani et al., 2017)
2. Transformer-RPR (Shaw et al., 2018)
3. Transformer-XL (Dai et al., 2019)
4. Adapted Transformer (Yan et al., 2019)

Compares models on three major tasks:

1. Performance Evaluation on Four NLP Classification Tasks (AG's News, Amazon Electronics, SST-2, SNLI)
2. Performance Evaluation on News Recommendation (MIND)
3. Regression Experiment on Amazon Ratings

Tables from: “DA-Transformer: Distance-aware Transformer”, Wu et Al, 2021

Performance

Methods	AG		Amazon	
	Accuracy	Macro-F	Accuracy	Macro-F
Transformer	93.01±0.13	93.00±0.13	65.15±0.40	42.14±0.41
Transformer-RPR	93.14±0.12	93.13±0.13	65.29±0.38	42.40±0.40
Transformer-XL	93.35±0.10	93.34±0.11	65.50±0.40	42.88±0.43
Adapted Transformer	93.28±0.13	93.27±0.14	65.47±0.39	42.69±0.42
*DA-Transformer	**93.72**±0.11	**93.70**±0.12	**66.38**±0.39	**44.29**±0.40

Table 3: Results on *AG* and *Amazon*. *Improvement over the underlined second best results is significant at $p < 0.05$.

Methods	SST		SNLI	
	Accuracy	Macro-F	Accuracy	Macro-F
Transformer	89.67±0.22	89.59±0.24	81.45±0.30	81.42±0.31
Transformer-RPR	89.94±0.19	89.90±0.20	82.20±0.31	82.18±0.31
Transformer-XL	90.06±0.20	90.02±0.21	83.19±0.29	83.15±0.30
Adapted Transformer	90.15±0.19	90.10±0.1	82.35±0.28	82.31±0.30
*DA-Transformer	**90.49**±0.17	**90.43**±0.19	**84.18**±0.27	**84.16**±0.29

Table 4: Results on *SST* and *SNLI*. *Improvement over the underlined second best results is significant at $p < 0.05$.

Methods	AUC	MRR	nDCG@5	nDCG@10
Transformer	67.76±0.18	33.05±0.16	35.94±0.19	41.63±0.20
Transformer-RPR	67.81±0.16	33.10±0.17	35.98±0.20	41.65±0.21
Transformer-XL	67.92±0.16	33.15±0.16	36.04±0.20	41.70±0.19
Adapted Transformer	67.70±0.22	33.01±0.20	35.89±0.17	41.58±0.23
*DA-Transformer	**68.32**±0.15	**33.36**±0.16	**36.34**±0.14	**42.07**±0.17

Table 5: Results on the *MIND* dataset. *Improvement over the underlined second best results is significant at $p < 0.05$.

DA-Transformer outperforms ALL previous major methods mentioned here

(1) Transformer (Vaswani et al., 2017)

(2) Transformer-RPR (Shaw et al., 2018)

(3) Transformer-XL (Dai et al., 2019)

(4) Adapted Transformer (Yan et al., 2019)

Key Concepts + Challenges

Introducing sparsity via RELU

Tunable sigmoid function for different ranges and distances

Different functions per head

Scaling values directly rather than introducing additional bias terms

Striving for mathematical clarity on the needs/ behaviour for Positional Encoding functions

Comprehensive performance audit on similar methods

A tunably flexible function still requires the learning process to capture it's range of expressiveness for maximum effect.

Introducing non-linearity in the middle of the attention block is a major departure from its intended use! - RELU LOSES INFORMATION

The presence of negative attention values in cells may not actually indicate grounds for exclusion in the context of a broader relationship in the network.

=> What do we want to leave up to training? What should remain fixed?

Positional Encoding: Modern Era

2021-2025

High Level Map

Treats positional encoding as the central lever for length extrapolation

Large research concern around the Context-Length Problem

Decoder architectures centralized around RoPE and its extensions (XPOS, PI, NTK-aware scaling, YaRN, LongRoPE, etc.)

Stronger theoretical and multimodal framing via surveys and 2D/3D relative encodings.

Explosion of multi-modal and domain-specific schemes that started to emerge in 2020

Rise of Fixed Positional Encodings

RoPE - Rotational Positional Encoding

Su et Al (2021)

RoPE - Rotational Positional Encoding (2021)

Asks a slightly different question than DA-Transformer.

If DA-Transformer asks:

What does an *optimal* Positional Encoding function look like?

RoPE asks:

What does the *minimal* Positional Encoding function look like?

RoPE - Why might we favor a minimal function?

RoPE offers a, fixed, predictable signal which doesn’t depend any parameters.

It encodes a function which captures the relative distance between elements.

RoPE does not manually influence the signal in a monotonic way, but with a long-term decay of values.

Lack of assumptions may enable superior reusability!

Figure from: “RoFormer: Enhanced Transformer with Rotary Position Embedding”, Su et Al, 2021

RoPE - Intuition

At a high level, RoPE functions by simply rotating the embedding representations of the tokens.

These rotations scale tokens with respect to their position.

Via trigonometric properties this will encode relative position differences in the dot products during self attention.

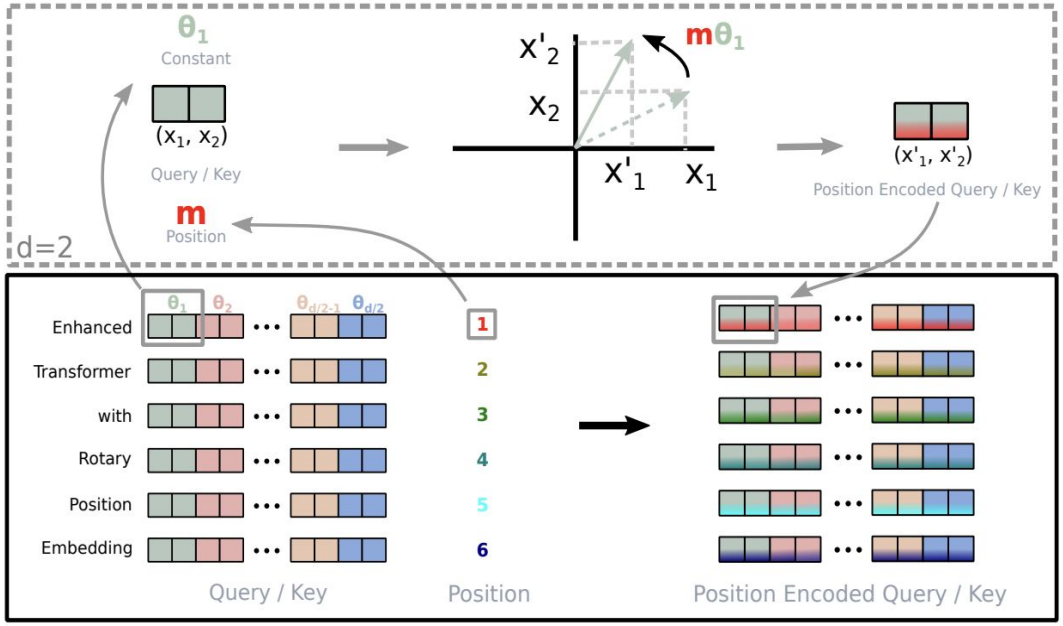


Figure 1: Implementation of Rotary Position Embedding(RoPE).

Figure from: “RoFormer: Enhanced Transformer with Rotary Position Embedding”, Su et Al, 2021

RoPE - Rotational Matrices

Mathematically speaking, the key behind RoPE is the use of rotation matrices, however, as these are sparse and inefficient to compute directly, a more efficient implementation is used which performs the same changes, but via two products.

$$\boldsymbol{R}^{d}_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

Rotational Function: Efficient Implementation

$$\text{Let } X, X', Cos, Sin \in \mathbb{R}^{N \times D},$$

$$R_\theta(X) = X \odot Cos + X' \odot Sin$$

Most implementations tend to formulate the rotation as a function that is simply directly applied to the Q and K tensors (sometimes V as well).

This function encodes absolute position information

Relative distance information becomes encoded during the attention dot products.

Rotational Function: Tools

$$\text{Let } X, X', Cos, Sin \in \mathbb{R}^{N \times D},$$

$$Cos = \begin{bmatrix} \cos(c_1\theta_1) & \cos(c_1\theta_1) & \cos(c_1\theta_2) & \cos(c_1\theta_2) & \dots & \cos(c_1\theta_{d/2}) & \cos(c_1\theta_{d/2}) \\ \cos(c_2\theta_1) & \cos(c_2\theta_1) & \cos(c_2\theta_2) & \cos(c_2\theta_2) & \dots & \cos(c_2\theta_{d/2}) & \cos(c_2\theta_{d/2}) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \cos(c_N\theta_1) & \cos(c_N\theta_1) & \cos(c_N\theta_2) & \cos(c_N\theta_2) & \dots & \cos(c_N\theta_{d/2}) & \cos(c_N\theta_{d/2}) \end{bmatrix}$$

This formulation encodes absolute position information through the scaling constants on the internal angles - c.

Note that this is a similar formulation as used in Vishwani et al.

Rotational Function: Tools

$$\text{Let } X, X', Cos, Sin \in \mathbb{R}^{N \times D},$$

X' is a reordering of the the embedding columns in order to align them to appropriate positions.

This new order reflects how they would correspond to the sin terms in a rotation matrix

In the original RoFormers paper, X' was formulated as shown.

There are other common implementations.

$$X'_i = \begin{bmatrix} -X_{i,2} \\ X_{i,1} \\ -X_{i,4} \\ X_{i,3} \\ \vdots \\ -X_{i,D} \\ X_{i,D-1} \end{bmatrix}$$

Efficient RoPE Formulation

$$\hat{\omega} = R_\theta(Q)R_\theta(K)^T = (Q \odot Cos + Q' \odot Sin)(K \odot Cos + K' \odot Sin)^T$$

$$\hat{\omega}_{[i,j]} = R_\theta(Q)R_\theta(K)^T_{[i,j]} = [(Q \odot Cos + Q' \odot Sin)(K \odot Cos + K' \odot Sin)^T]_{[i,j]}$$

As with the sinusoidal encoding of Vishwani, specifically the Pos-Pos term, further evaluation reveals a linear combination of products between sin and cos waves.

This resolves into relative differences in angles via trigonometric properties.

Thus, relative distance are encoded as each angle corresponds to token position.

Cell-Level Expansion

$$\hat{\omega}_{[i,j]} = [(Q \odot Cos + Q' \odot Sin)(K \odot Cos + K' \odot Sin)^T]_{[i,j]}$$

$$= (Q \odot Cos)(K' \odot Sin)^T_{[i,j]} + (Q \odot Cos)(K \odot Cos)^T_{[i,j]}$$

$$+ (Q' \odot Sin)(K' \odot Sin)^T_{[i,j]} + (Q' \odot Sin)(K \odot Cos)^T_{[i,j]}$$

$$= (Q \odot Cos)_{[i]} \cdot (K' \odot Sin)_{[j]} + (Q \odot Cos)_{[i]} \cdot (K \odot Cos)_{[j]}$$

$$+ (Q' \odot Sin)_{[i]} \cdot (K' \odot Sin)_{[j]} + (Q' \odot Sin)_{[i]} \cdot (K \odot Cos)_{[j]}$$

RoPE - Game Changing Impact

RoPE's minimalism, effectiveness, and rapid adoption into many Decoder-Only architectures quickly earned it a dominant spot within the sub-field of PE.

RoPE has found an early adopter in LLaMa 2 and has been widely used since in many major decoder-only LLM families:

- LLaMa
- Mistral
- Falcon
- Qwen
- Gemma
- DeepSeek

RoPE - Extensions

RoPE tends to struggle to capture extremely long distances.

This has led to a variety of methods used for extending RoPE based models:

- YaRN
- LongRoPE
- NTK-Aware Scaling
- XPos

Context Length Problem

Core problem:

Transformers do not inherently know what position a token is in—unlike RNNs, which move step-by-step.

We inject position using a chosen positional encoding (PE).

This encoding is usually designed for a fixed maximum length, like 512 or 2048 tokens.

But LLMs want to reason across: 8k, 32k, 128k, 1 million tokens

So the question becomes:

If the model only ever saw sequences of length N during training, can it still reason correctly about sequences much longer than N?

This is **length extrapolation**.

Why is this more important now?

Between 2018–2021, 512 tokens felt big enough.

By 2024–2025, LLM users expect:

128k context for agent memory

1M+ tokens for codebases

entire books as input

multi-week conversation history

cross-modal sequences (video frames, audio chunks, actions)

This created a bottleneck:

The positional encoding, not the architecture, became the limiting factor.

Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation

Press et Al (2022)

ALiBi: Motivations and Design

The primary research questions motivating ALiBi:

Can Transformers extrapolate to longer input sequences than those seen in training?

Is the positional encoding method the root cause of extrapolation failure?

Can changing the position representation alone enable extrapolation?

Can extrapolation be achieved efficiently, without extra parameters, slower runtime, or higher memory?

ALiBi: Intuitive View

Leverages relative positional encoding

Adds it’s encoding in a single fixed bias term

Uses a different bias value for each head in the attention mechanism

Each bias term is a direct rescaling of a relative position matrix, with each head changing the slope that is applied

Cell-Level View:

$$\text{Let } m \in \mathbb{R}^{H \times N \times N} \text{ s.t. } m_{[h]} = \left(\tfrac{1}{2}\right)^{\frac{8h}{H}} R.$$
$$R \in \mathbb{R}^{N \times N} \text{ and } R_{[i,j]} = i - j,$$

$$\hat{\omega}_{[h,i,j]} = (UW_q)(UW_k)^T_{[h,i,j]} - \left(\frac{1}{2}\right)^{\frac{8h}{H}} R_{[i,j]} = Q_{[h,i]} \cdot K_{[h,j]} - \left(\frac{i-j}{2}\right)^{\frac{8h}{H}}$$

Designed as a relative positional encoding method to offer greater extrapolation capabilities then contemporary approaches: T5, RoPE.

Uses a different signal for each attention head.

ALiBi is parameterless, fixed signal based on relative distances between tokens.

ALiBi encodes a monotonic decay with respect to increasing distance at different rates in each attention head.

Tensor-Level View:

$$\text{Let } m \in \mathbb{R}^{H \times N \times N} \text{ s.t. } m_{[h]} = \left(\tfrac{1}{2}\right)^{\frac{8h}{H}} R.$$
$$R \in \mathbb{R}^{N \times N} \text{ and } R_{[i,j]} = i - j,$$

$$\hat{\omega} = (UW_q)(UW_k)^T - m$$

Using a relative position matrix or mask/buffer we can easily reflect the previous expression in the form of a static bias term during a high level formulation.

The above definition for m effectively encodes a monotonic decay with respect to increasing distance at different rates in each attention head.

Experimentation: Motivation

The paper is motivated by one central problem in Transformers:

Transformers trained on short sequences fail catastrophically when asked to process longer sequences at inference.

Plan:

Train on short sequences, evaluate on much longer ones (e.g., train on 512, test on 4096).

Experimentation: Tasks

TASK 1 — WikiText-103 (Autoregressive LM)

- Predict next token given prior tokens.
- Evaluate perplexity on sequence lengths much longer than training.
- Used to examine catastrophic extrapolation in sinusoidal/RoPE/T5 biases.

TASK 2 — Long-sequence extrapolation tests

- Train on L = 512, 1024, 2048
- Evaluate on Lvalid up to 16k tokens
- Compare ALiBi vs sinusoidal, ROPE, learned absolute, T5 relative bias

TASK 3 — Domain transfer on BookCorpus

- Train slopes only on WikiText-103
- Evaluate on BookCorpus without re-tuning
- Tests robustness across domains

Experimentation: Metrics

Perplexity (PPL) - Lower is better.

Used for:

- WikiText-103
- BookCorpus
- CC100/RoBERTa large-scale experiment

Experimentation: Primary Findings

ALiBi generalizes to longer sequences without degradation, while all baselines fail.

ALiBi improves in-domain LM perplexity even when not extrapolating.

Training shorter sequences with ALiBi can beat longer-sequence baselines.

Slopes generalize across domains.

ALiBi generalizes to longer sequences without degradation, while all baselines fail.

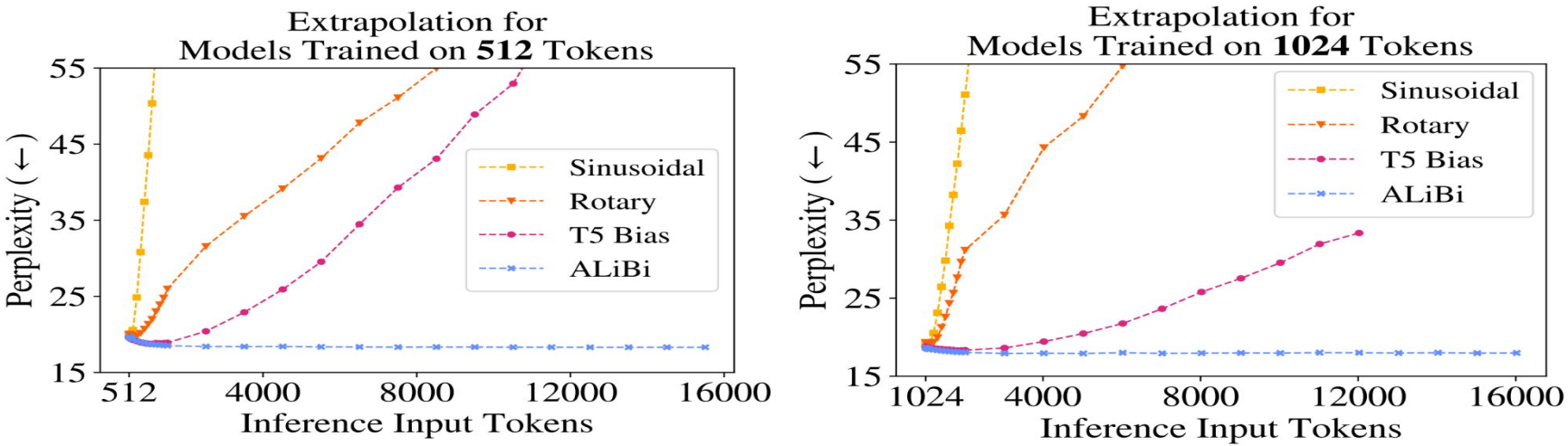


Figure 1: Extrapolation: as the (validation-set's) input sequence gets longer (x-axis), current position methods (sinusoidal, rotary, and T5) show degraded perplexity (y-axis, lower is better), but our method (§3) does not. Models were trained on WikiText-103 with sequences of $L = 512$ (left) or $L = 1{,}024$ (right) tokens. T5 ran out of memory on our 32GB GPU. For more detail on exact perplexities and runtimes, see Tables 2 and 3 in the appendix.

Figure from: “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation”, Press et Al, 2022

Training shorter sequences with ALiBi can beat longer-sequence baselines.

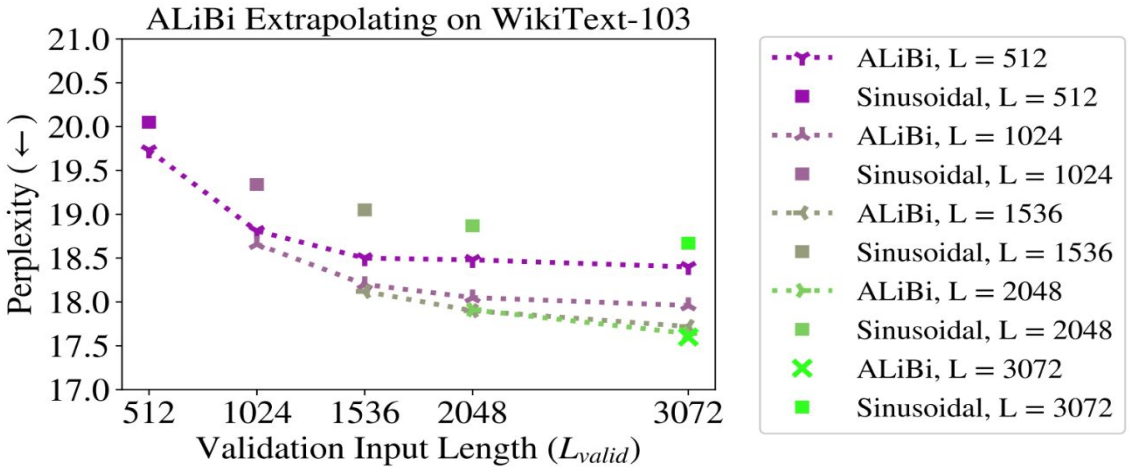


Figure 4: ALiBi models trained and evaluated on varying sequence lengths on the WikiText-103 validation set and the sinusoidal baseline (not evaluated on longer sequences). All of our models outperform the sinusoidal ones even when trained on fewer tokens. Appendix Table 5 has exact perplexities, more ALiBi models (trained on fewer tokens), and results for rotary and T5 bias models.

Figure from: “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation”, Press et Al, 2022

Training shorter sequences with ALiBi can beat longer-sequence baselines.

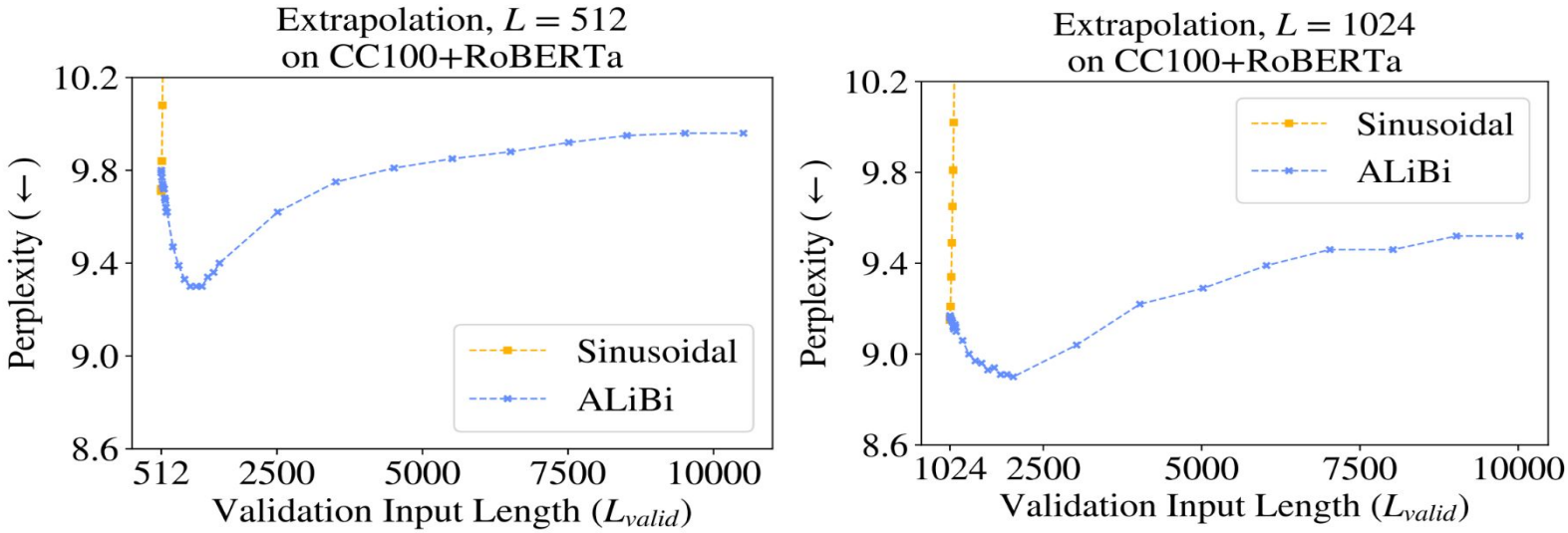


Figure 6: The ALiBi and sinusoidal models (with both $L = 512$ and 1024) trained for 50k updates (1 epoch) on the CC100+RoBERTa corpus, extrapolating on the validation set. ALiBi achieves the best results at around $2L$ but maintains strong performance even up to 10000 tokens in these experiments.

Figure from: “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation”, Press et Al, 2022

ALiBi improves in-domain LM perplexity even when not extrapolating

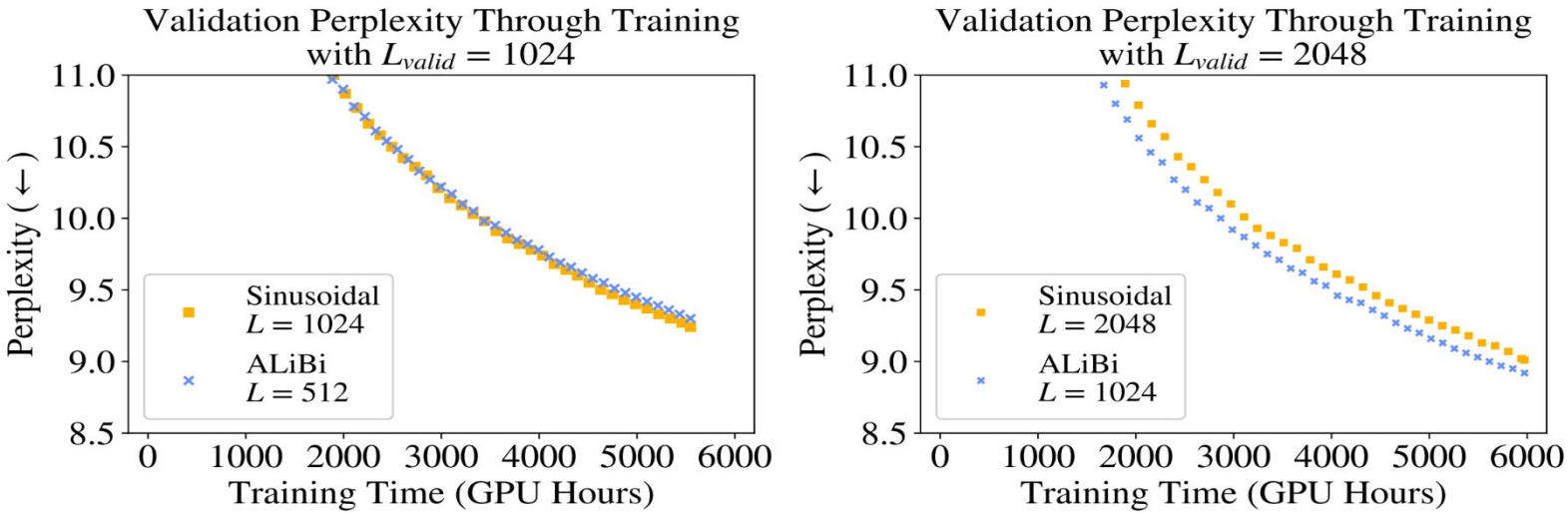


Figure 5: On the left (right), a 1.3B-parameter ALiBi model trained on 512 (1024) and evaluated on 1024 (2048) tokens during training, compared to the sinusoidal baseline trained on 1024 (2048) tokens. The ALiBi models obtain strong results even though they use 6%-11% less memory since they train on shorter sequences. Appendix Table 11 shows memory use and end-of-training perplexities.

Figure from: “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation”, Press et Al, 2022

ALiBi’s use for Fixed Parameters improves speed and Memory Efficiency

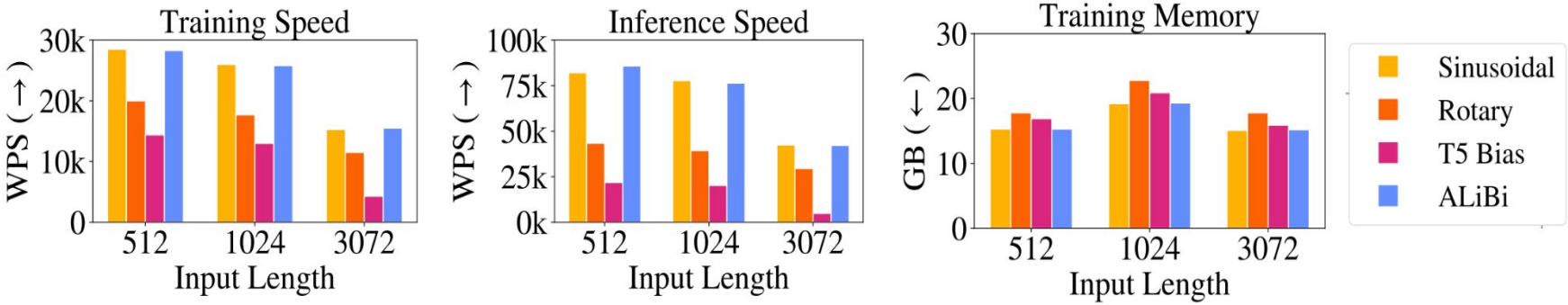


Figure 2: A comparison of batched training, inference speed and memory use of the sinusoidal, rotary, T5 bias, and our ALiBi position methods. The speed differences between our method and the sinusoidal are within 1% during training and 3% for inference, which is insignificant on our hardware. ALiBi uses 100MB of extra memory when training on input lengths 1024 and 3072 in this setting. Memory usage is lower in all approaches when training on 3072 tokens (compared to 1024) since we break batches into multiple updates. See Table 1 in the appendix for exact numbers.

ALiBi Takeaways

Experimentation shows that positional encoding is largely responsible for extrapolation capabilities

Carefully chosen fixed Positional signals can outperform learnable encoding schemas in next token prediction tasks and exhibit much superior extrapolation.

Applying different slopes per head enables learning attention scores at different ranges between to elements

Using a fixed signal enables a predictable signal of new tokens.

Valuable for generative models who encounter large sequences unseen in training.

NoPE

(202X?)

What is NoPE? - No Positional Encoding!

The architecture still encodes order, but via:

- constrained attention masks (causal, banded, local)
- parameter sharing across layers
- convolutional front-ends
- state-space dynamics (SSMs)
- learned recurrences
- structural biases in attention weights

How did this Idea Emerge?

Position Information Leaks into Attention Weights Naturally

If a transformer:

- uses causal masking
- and shares weights across layers

…then early layers learn token-distance patterns automatically.

Figure from: “The Impact of Positional Encoding on Length Generalization in Transformers”, Kazemnejad et Al, 2023

How does NoPE Perform?

Surprisingly well!... Given the set up

Several papers document superior generalization within certain tasks and extrapolation ranges.

Looking at these results, it seems almost seems promising to abandon encoding all together!

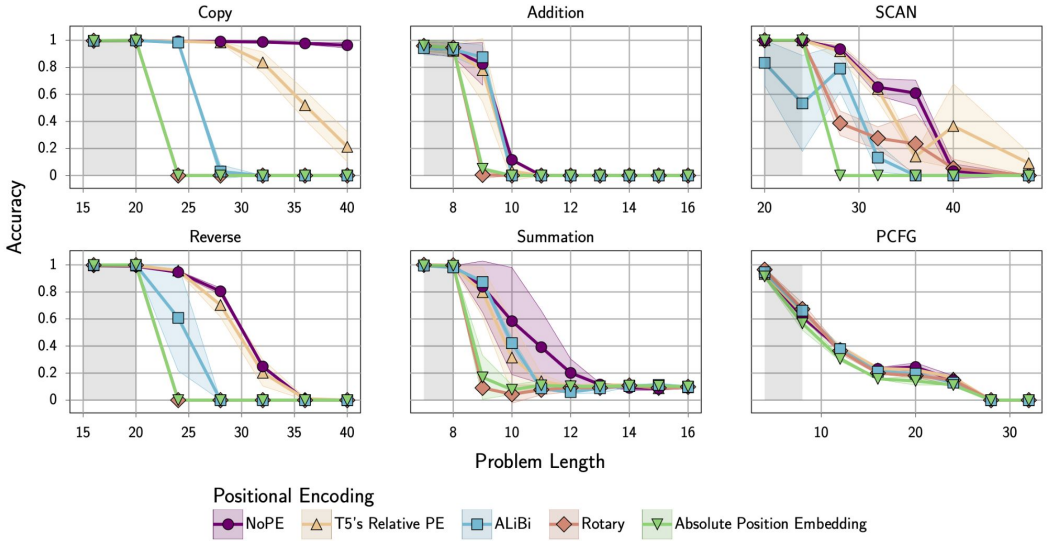


Figure 3: Showcasing the generalization behavior of different positional encodings on 6 datasets. The shaded area represents evaluation examples with I.I.D. lengths (i.e. seen during training). Since all models perform perfectly, or close to it, on the I.I.D. lengths (measured on unseen examples), for improved readability, we only show a subset of them in the figure. Refer to Appendix E for more detailed plots.

Is NoPE an Existential Threat to Positional Encoding?

NOPE! :D

Why?

Largely only applies to decoder-style or hybrid models:

Encoder-Style models primarily rely on bi-directional attention.

Other Limitations of NoPE?

NoPE also fails at:

1. Long-context extrapolation ***

Without a formal mechanism like RoPE/ALiBi, models degrade beyond training length after a certain point.

2. Multimodal input

Images, video, audio, and spatial reasoning require explicit spatial coordinates.

3. Bidirectional encoders

Encoders with full attention but no causal mask lack any ordering constraint unless PE is provided.

4. Structural tasks

Code, graphs, trees, tables require positional clues.

Major Caveat ***

The tasks in this paper use:

- Context lengths ≤ 40–50 tokens
- Clean synthetic symbolic data (Copy, Reverse, Addition, SCAN, PCFG)
- Autoregressive decoder-only models where causal masking already gives an ordering signal

In this regime:

- The model can infer relative order directly from the structure of the computation.
- Length generalization means going from e.g. 20 → 40 tokens.
- In this narrow setting, NoPE implicitly learns relative position patterns (they even prove this theoretically), and outperforms all explicit PEs.

But this is not long-context extrapolation in the usual LLM sense.

Where is NoPE Viable?

Some architectures don’t need PE at all!

Hybrid Models: (i.e. RWKV)

A hybrid RNN–Transformer with no explicit positional encoding; sequence order is embedded through gated recurrent dynamics.

Mamba / S4 / State-Space Models:

- Continuous-time system evolves hidden state with time-dependent kernels.
- Positional order emerges from discretization of the ODE / convolution kernels.

Local Attention / Convolutional front-ends

Models like Sparse Transformers, LongNet, Hyena can implicitly encode locality / distance via convolutional or dilated kernels.

Rise of Multimodal and Domain-Specific Needs

Started to emerge in force around 2020-2021

Vision Transformers (2D positional geometry)

ViT Dosovitskiy et al. 2020

Encoding type: Learned 2D absolute grid

Why it matters:

First major ViT; patches get absolute 2D embeddings.

No extrapolation — shows limitations of learned absolute PEs.

DeiT / Swin Transformers (2021)

Encoding type: Hierarchical & relative 2D biases

Why it matters:

Swin builds local windows, using relative positional bias in 2D.

This becomes the dominant method in vision backbones.

DETR / Deformable DETR (2020–2021)

Encoding type: Sine-cosine 2D + learned offsets

Why it matters:

Proved that detection requires absolute spatial coherence, not just sequence order.

2D sinusoidal PEs become standard in object detection.

BEiT-BERT for Images (2021–2022)

Encoding type: Relative 2D biases for image masking

Why it matters:

Brings NLP-style relative PEs directly into masked image modeling.

Video Transformers (3D or Spatiotemporal PEs)

Video = time × height × width → PEs must encode 3D grids.

TimeSformer — Bertasius et al. 2021

Encoding type: Separable temporal PE + spatial PE

Why it matters:

Established the standard:

PE(time) + PE(height) + PE(width), sometimes with RoPE variants later.

ViViT — Arnab et al. 2021

Encoding type: Factored 1D temporal + 2D spatial PEs

Why it matters:

Demonstrates multiple PE factorizations for video-recognition tasks.

Video Masked Autoencoders (2022–2024)

Encoding type: 3D sinusoidal / 3D learned / 3D relative

Why it matters:

Video MAEs reveal that temporal positional stability matters greatly for reconstruction.

Video LLaMA, PaLI-X, Flamingo (2022–2024)

Encoding type:

- 2D vision PEs for frames
- 1D temporal PEs to order frame sequences

Cross-modal alignment embeddings

Audio / Speech Transformers (high-res temporal PEs)

Audio is a very long 1D signal with sampling rates (16k–48k Hz) → huge positional spans.

wav2vec 2.0 (2020) / HuBERT (2021)

Encoding type:

Convolutional downsampling → positional info emerges through receptive fields

(not explicit PEs)

Why it matters:

Showed that CNN front-ends can act as implicit positional encoders.

SpeechT5, AudioLM, Whisper (2022–2023)

Encoding type:

Mostly sinusoidal or relative bias-based PEs post-CNN tokenizer.

Why it matters:

Whisper (2022) uses absolute learned PEs but relies heavily on strided downsampling, showing an alternate path to long-context robustness.

Audio Spectrogram Transformers (AST, 2021–2024)

Encoding type: 2D PEs (freq × time)

Why it matters:

Treats audio as an image; requires 2D RoPE or 2D bias for cross-frequency coherence.

Molecules, Proteins, and 3D Scientific Geometry

These papers **highlight graph distance, 3D coordinates, and rotational invariance** as forms of positional information.

SE(3)-Transformers, Fuchs et al. 2020

Encoding type:

- Position = 3D spatial coordinates
- PE derived from group theory (SE(3)-equivariance)

Why it matters:

One of the first to treat positional encoding as geometric invariance instead of sequence order.

Graphormer Ying et al. 2021

Encoding type:

- Graph shortest-path distance
- Centrality encodings
- Edge features

Why it matters:

Redefines “position” as topological distance on a graph.

This becomes standard in molecule/property prediction pipelines.

Geoformer & Protein RoPE Variants (2023–2025)

Encoding type:

3D relative position kernels, angle encodings, and pairwise distances.

Why it matters:

Apply RoPE-like ideas to Euclidean and spherical manifolds.

A direct continuation of your “geometry-aware RoPE” theme.

Climate, Physics, Simulation, and Spatiotemporal Data

FourCastNet (NVIDIA 2022)

Encoding type:

- 2D Earth-surface grid encoded via spherical coordinates (lat/lon)
- Often augmented with Fourier features.

Why it matters:

Shows positional encoding as physical coordinate encoding.

EarthFormer / ClimateBERT (2022–2024)

Encoding type:

- 3D grids (lat × lon × time)
- Learned geodesic PEs
- Rotary 2D + temporal RoPE

Why it matters:

Establishes PEs for global, geophysical simulation data.

Neural Operators (Fourier Neural Operator, FNO; 2020–2023)

Encoding type:

Coordinates encoded as Fourier positional features

Why it matters:

Brings continuous positional encoding from scientific computing into deep learning.

MeshGraphNets (2020–2022)

Encoding type:

Graph structures with relative position and geometric distances.

Why it matters:

Physical simulations rely heavily on PE to encode the mesh structure itself.

General Multimodal Foundation Models (2022–2025)

Flamingo — DeepMind 2022

Encoding type:

Vision: 2D learnable/bias PEs

Text: 1D RoPE/sinusoidal

Cross-modal attention uses special learned positional bridge

Significance:

Early and influential example of positional unification in multimodal models.

PaLI / PaLI-X (2022–2023)

Encoding type:

Explicitly align 2D vision and 1D language PEs.

LLaVA (2023–2024)

Encoding type:

Use CLIP’s 2D PEs and map them into LLaMA’s 1D RoPE space.

Common Concepts and Goals in Positional Encoding

Extracting a Toolkit

Sinusoidal Waves

Parameter Free

Encodes absolute position information directly

Relative distances emergently via dot product

Supports Directionality (TENER)

Enables rotational encodings (RoPE)

Ubiquitous due to Vishwani and RoPE

Highly represented in open source decoders

Extrapolation is somewhat limited when used alone (ALiBi > raw RoPE)

Often in practice combined with a means of improving extrapolation (i.e. YaRN, etc.)

Not as well represented in Encoder-style models

Relative Position Matrices

$$R_{[i,j]} = |i - j|$$

Easily integratable in to bias terms for the attention scores…

Combined with some additional treatment to be applied to the scores themselves:

- Scaling
- Custom Functions (DA)
- Associated with learnable parameters (Shaw, Huang, etc.)

Key Problem: Quadratic Space Usage!

Many implementations try to find efficient lookup work arounds (i.e. Shaw et Al)

Still used directly in some cases as well

Hadamard Products

Broadly used for mixing and gating signals in Deep Learning

Commonly combined with relative-position tensors for scaling with functions on token-by-token distances

DA-Transformer combines the Hadamard Product with an activation function to scale only desirable signals (adding here would not work)

Also used in RoPE implementations

Learnable Absolute Position Parameters

More popular in early research for text transformers, out of favor now

Used in some vision transformers

Still used in text Encoder-only transformers (BERT)

Benefits from smaller and fixed context lengths

Occasionally used in time-series applications

Fixed (Parameterless) Positional Signals

Why use?

Boosts extrapolation via predictable signals

- ALiBi

Combinable with learnable parameters

- TENER

Encodes Absolute and Relative Information

- Sinusoidal
- RoPE

When not to use?

Fixed or small context windows

Ideas from Multi-Modal Research

Image

2D Absolute Grids (ViT 2020)

2D Sinusoidal (DETR 2021)

Relative 2D Biases (BEiT 2022)

Video

Seperal Temporal + Spatial PE (TimeSformer)

3D Sinusoidal

2D Vision + 1D Temporal (2022-2024)

Audio

Convolutional downsampling (HuBERT 2021)

2D (Freq x Time) (2021-2024)

Foundation Models

2D Learnable Bias + 1D Text (Flamingo 2022)

Mapping 2D Biases into RoPE Space (LLaVA 2023)

Ideas from Domain-Specific Research

Scientific Geometry

Group Theoretic Encodings (2020)

Graph Distances w/ Edge Features (Graphformer 2021)

Centrality Measures (Graphformer 2021)

Physics/Simulation

2D Grid Encodings, via Spherical Lat/Lon (2022)

Lat-Lon-Time 3D PE (2022-2024)

Learned Geodesic PE (2022-2024)

Less Popular Ideas

Separate Positional and Content Parameters (XL, DeBERTa):

- Introduces greater overhead
- Content-position encoding remains effective
- Implementers also often incorporate additional mechanism to reach max effectiveness (XL,DeBERTa)

Same Parameters Across Layers (T5):

- Theoretically shaky: Should the same learnable signal be shared across hierarchies of abstraction?
- Very rare in practice

Partitioning the Embedding Space (Axial):

- Theoretically shaky: Embedding space is richest source of feature expression
- Axial remains one of extremely few methods to implement this

What is the Future of Positional Encoding?

What is the Future of Positional Encoding?

This question has different answers depending on if we are still referring to the transformer architecture, and if we are not.

If we are are referring to the transformer architecture, then we must look at it through the two major families:

1. Encoder
2. Decoder

What is the Future of PE in Transformers?

Encoder:

- Custom, highly tailored signals
- Domain oriented
- Emphasis on rich encodings
- Designed for smaller transformer models
- Mixture of learned, absolute, and relative signals are used
- Fixed, Smaller Contexts are common

Decoder:

Text-based models converged on RoPE

This often means use-case specific extensions

Vision Decoder-Transformers often still use learned absolute position embeddings

Encoder: Characterizing the Needs of the Field

No default Positional Encoding method dominates Encoder architectures

This is not surprising as the task range of Encoder-Transformers is diverse:

- Sentiment analysis
- Grammatical correctness
- NER: Named Entity Recognition
- Semantic Analysis
- Scientific and Engineering domains

Encoders tend to be smaller & geared towards a wide range of specific tasks

Reasons for Learned/Hybrid PE in Encoders

- **Fixed Sequence Lengths**: Intended for tasks with fixed maximum sequence lengths (typically 512 tokens), and they prioritize computational efficiency and strong in-domain performance.

- **Simplicity**: Learned absolute embeddings are simple to implement and computationally efficient because they involve a straightforward lookup and addition operation, which is fast during both training and inference within the defined maximum length.

- **Task-Specific Performance**: Models can learn highly effective, task-specific positional representations from the data, performance within its limits.

- **Bidirectional Context**: Encoder models use a bidirectional attention mask, meaning every token can attend to all other tokens in the sequence. Absolute positions help in explicitly defining the location of each token in this global context.

Decoder: Has a particular method (RoPE) “won”?

Yes and No.

RoPE has become the default method for a variety of reasons, based both directly upon its own high merits and those due to market forces.

Large, generative models benefit from common and transferable architectural designs. There is also heavy resource costs associated with hosting, pre-training, and altering these models.

The task range of Decoder-style models is also relatively homogenous.

All this creates an ecosystem that incentivizes a culture of adoption, extension, and repurposing rather than “bottom-up” approaches to design innovation.

Reasons for Relative Positional Methods in Decoders

- **Length Generalization**: Need to extrapolate to sequence lengths not encountered during training. Used for generative tasks where output length can vary greatly, and the model needs to handle very long prompts and generations.

- **Modeling Relative Distances**: Maintains a coherent context over thousands of tokens, as the relationship between nearby words is generally more important than the absolute position in the entire text.

- **Avoiding Overfitting to Position**: Learned absolute embeddings can sometimes overfit to specific training positions and struggle when the sequence length in testing exceeds the training maximum. RoPE and ALiBi inherently handle this challenge by using mathematical functions to encode relative positions, which generalizes better to longer contexts.

What does Research in PE look like in 2025?

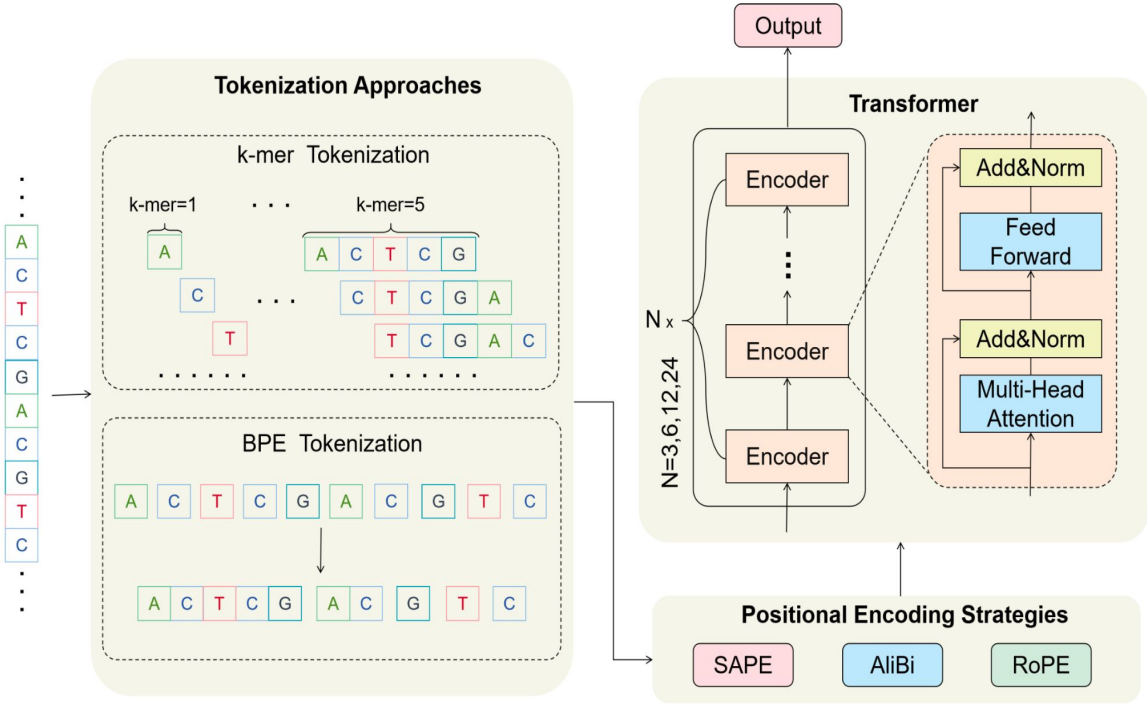
Hint: Pure AND Applied

Evaluation of Coding Schemes for Transformer-based Gene Sequence Modeling

Gong et Al (2025)

Figure from: “Evaluation of Coding Schemes for Transformer-based Gene Sequence Modeling”, Gong et Al, 2025

Applied Findings



Findings show that different positional encodings schemes capture different properties (i.e.):

RoPE excels at capturing periodic motifs and extrapolating to long sequences.

AliBi also performs well on tasks driven by local dependencies.

What does this remind us?

Positional Encoding methods are diverse mathematical signals.

Different methods carry different profiles and effects, not just scores on performance benchmarks to be uncritically compared.

As transformers continue to diffuse into greater and greater niches of domain-specific research, there will be both qualitative and empirical needs driving application and innovations.

Tab-PET: Graph-Based Positional Encodings for Tabular Transformers

Leng et Al (2025)

Figure from: “Tab-PET: Graph-Based Positional Encodings for Tabular Transformers”, Leng et Al, 2025

Pure Research Questions

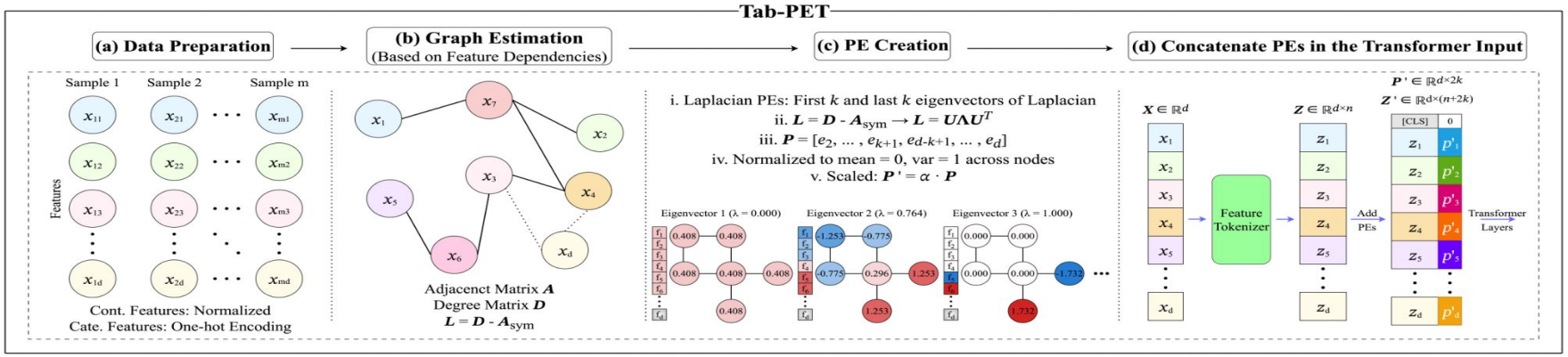


Figure 1: Tab-PET framework for integrating PEs in tabular transformers. (a) Categorical features are one-hot encoded and continuous features are normalized. (b) A feature-wise graph is estimated based on intra-feature dependencies, capturing relational structure among dimensions. (c) Graph Laplacian eigenvectors (examples shown) are extracted to form fixed PEs and scaled using the hyperparameter α to emphasize the degree of importance. (d) These encodings are concatenated with standard embeddings and fed into transformer layers.

How can the structural cues unique to tabular data be encoded into the transformer architecture in a way analogous to positional encoding?

What does this remind us?

Positional Encoding methods are still largely based on the original needs of NLP

However, transformers have spread to various domains and now interact with a diverse array of data types.

Certain data types require different treatments than those originating, or derived from, text data, and assumptions from Natural Language.

Why this all Matters Beyond Transformers

(My Soapbox)

Personal Soapbox: Tools Outlive Models

Positional Encoding is a means by which we can reflect structural cues, or concepts of position, directionality, or distance via signals in an optimization friendly way.

It enables a separation of concerns with architectures to optimize without inherently having to accommodate these concepts themselves.

These methods are also intended to be injectable to architectures which are agnostic to these concepts as well.

Positional Encoding is now largely still seen as means to an end, rather than a vehicle to enrich existing architectured or empower future ones.

Final Takeaways

Decoder-Transformers have converged around RoPE and extensions

Encoder-Transformers still have an active avenue of domain-specific formulations for positional encoding methods

Saturation is occurring in a limited capacity, and only in certain parts of NLP

Future Research in Positional Encoding will be both Pure AND Applied:

- Applied: Biology, Physics, Time-Series, Engineering (etc.) Specific-Problems
- Pure: Novel Data Types, Reflecting Higher Structural Abstractions

Positional Encoding is a mathematically rich tool rather than a means to an end

Thank You!

Christopher Curtis

Bonus Slides - Datasets

SQuAD v1.1

Type: Extractive Question Answering

What the data looks like:

- A paragraph from Wikipedia
- A question about the paragraph
- A span of text within the paragraph that is the correct answer

Example:

> “Where was Marie Curie born?”

…the model must look at the paragraph, locate the exact span of text, and return something like:

> “Warsaw, Poland”

SQuAD v1.1

Type: Extractive Question Answering

What the data looks like:

- A paragraph from Wikipedia
- A question about the paragraph
- A span of text within the paragraph that is the correct answer

Example:

> “Where was Marie Curie born?”

…the model must look at the paragraph, locate the exact span of text, and return something like:

> “Warsaw, Poland”

GLUE Benchmark - Huang et Al 2020

Includes:

MNLI (matched/mismatched) → natural language inference

QQP → paraphrase identification (F1)

SST-2 → sentiment classification

MRPC → paraphrase classification (F1)

MNLI – Multi-Genre Natural Language Inference

Data:

Pairs of sentences from multiple genres (fiction, government, spoken dialogue, travel guides, etc.)

Task:

Determine whether the second sentence is:

- Entailed by the first (logically follows)
- Contradicted by the first (logically impossible)
- Neutral (neither follows nor contradicts)

MNLI – Multi-Genre Natural Language Inference

Example:

Premise: “A man is playing a guitar on stage.”

Hypothesis: “A musician is performing.”

➡ Entailment

What MNLI tests:

- Deductive reasoning
- Semantic equivalence
- Real-world knowledge
- Understanding negation, modality, and uncertainty

QQP – Quora Question Pairs

Data:

Two questions posted on Quora.

Task:

Determine if both questions ask the same thing, even if phrased differently.

QQP – Quora Question Pairs

Example:

“How can I lose weight fast?”

“What are quick ways to burn fat?”

➡ Duplicate

What QQP tests:

- Paraphrase detection
- Semantic similarity
- Robustness to lexical variation
- Rewriting patterns (“lose weight” ↔ “burn fat”)

SST-2 – Stanford Sentiment Treebank (Binary Sentiment)

Data:

Single sentences from movie reviews.

Task:

Classify the sentiment as positive or negative.

Example:

“This film was an absolute triumph.”

➡ Positive

What SST-2 tests:

- Emotional valence
- Polarity cues (“but”, “although”, “barely”)
- Subtle sentiment patterns (“surprisingly decent” is positive)

MRPC – Microsoft Research Paraphrase Corpus

Data:

Pairs of news sentences referring to the same events or facts.

Task:

Predict whether the two sentences mean the same thing semantically.

Example:

“The CEO resigned yesterday.”

“Yesterday, the company’s chief executive stepped down.”

➡ Paraphrase

What MRPC tests:

- Deep paraphrase reasoning
- Handling syntax shifts (“resigned“ ↔ “stepped down”)
- Entity matching (“CEO” ↔ “chief executive”)

CoLA – Corpus of Linguistic Acceptability

Data:

Short sentences labeled as grammatically acceptable or unacceptable based on linguistic judgment.

Task:

Decide whether a sentence is grammatically correct.

Example:

“The dog chased the cat.” → Acceptable

“Chased the dog the cat.” → Unacceptable

What CoLA tests:

- Syntax
- Grammar generalization
- Ability to understand structural well-formedness

QNLI – Question-answering NLI task (from SQuAD)

Data:

A question + a sentence extracted from the paragraph of SQuAD.

Task:

Predict whether the sentence contains the answer.

Example:

Question: “When was the Eiffel Tower built?”

Sentence: “Construction began in 1887.”

➡ Answerable

What QNLI tests:

- Semantic matching
- Reasoning about whether text could contain an answer
- Local relevance detection

AG’s News (AG News Classification)

Type: Topic Classification

What the data contains:

News headlines + short descriptions

4 categories: World, Sports, Business, Sci/Tech

What the model must do:

Read a short news text and predict which category it belongs to.

What skills this tests:

- Coarse-grained topic recognition
- Lexical cues (“stocks”, “NASA”, “tournament”)
- Medium-length document encoding

Amazon Electronics Review Polarity (Amazon Review Dataset)

Type: Sentiment / Polarity Classification

What the data contains:

User reviews about electronics

Labels are positive or negative

What the model must do:

Read a product review and classify sentiment.

What skills this tests:

- Polarity understanding
- Handling informal, noisy text
- Long sentences and mixed sentiment (“good value but poor build”)

SNLI (Stanford Natural Language Inference)

Type: Natural Language Inference (Entailment)

What the data contains:

Pairs of short sentences

Labels: entailment, contradiction, neutral

What the model must do:

Determine if the hypothesis is implied by, contradicts, or is unrelated to the premise.

What skills this tests:

- Logical reasoning
- Paraphrase vs contradiction
- Negation understanding
- Entity tracking

MIND (Microsoft News Recommendation Dataset)

Type: Personalized News Recommendation

What the data contains:

- User click history
- Candidate news items to rank
- News titles & abstracts

What the model must do:

Predict which news articles a user is likely to click.

Skills this tests:

- User preference modeling
- Relevance ranking
- Matching candidate documents with user history

Amazon Rating Regression (Amazon Product Ratings)

Type: Regression (Predicting 1–5)

What the data contains:

Product reviews (1-star to 5-stars)

What the model must do:

Predict the exact numeric rating rather than polarity.

Skills this tests:

- Fine-grained sentiment
- Handling subtle tonal shifts
- Regression modeling (continuous output)

WikiText-103 (Language Modeling)

What the data contains:

- ~103M tokens of English Wikipedia text
- Long-form, natural English paragraphs
- A standard open benchmark for autoregressive LMs

What the model must do:

Predict the next token given all previous tokens

Why WikiText-103?

It has very long sequences, making it ideal for measuring:

- Extrapolation beyond training length
- Sensitivity to positional encoding

Evaluation metric:

Perplexity (lower = better)