Exploring the Effectiveness of Positional Embedding on Transformer-Based Architectures for Multivariate Time Series Classification

Chao Yang(✉), Yakun Chen, Zihao Li, and Xianzhi Wang

School of Computer Science, University of Technology Sydney, Ultimo, Australia
chao.yang@student.uts.edu.au

Abstract. Positional embedding is an effective means of injecting position information into sequential data to make the vanilla Transformer position-sensitive. Current Transformer-based models routinely use positional embedding for their position-sensitive modules while no efforts are paid to evaluating its effectiveness in specific problems. In this paper, we explore the impact of positional embedding on the vanilla Transformer and six Transformer-based variants. Since multivariate time series classification requires distinguishing the differences between time series sequences with different labels, it risks causing performance degradation to inject the same content-irrelevant position token into all sequences. Our experiments on 30 public multivariate time series classification datasets show positional embedding positively impacts the vanilla Transformer's performance yet negatively impacts Transformer-based variants. Our findings reveal the varying effectiveness of positional embedding on different model architectures, highlighting the significance of using positional embedding cautiously in Transformer-based models.

Keywords: Positional embedding · Multivariate time series classification · Deep learning

1 Introduction

Multivariate time series classification plays a critical role in various fields, such as gesture recognition [29], disease diagnosis [15], and brain-computer interfaces [6]. Recent years have witnessed Transformer-based methods making remarkable breakthroughs in numerous disciplines, such as natural language processing [19,24], computer vision [1,9], and visual-audio speech recognition [20,22]. This success has inspired an increasing application of Transformer-based architectures [4,16,30] to multivariate time series classification. Besides, Transformer's ability to perform parallel computation and leverage long-range dependencies in sequential data make it especially suitable for modeling time series data [26].

Since Transformer is position-insensitive, positional embedding was introduced to allow the model to learn the relative position of tokens. Positional embedding generally injects position information into sequence data [23]. It takes

X. Yang et al. (Eds.): ADMA 2023, LNAI 14176, pp. 34–47, 2023.
https://doi.org/10.1007/978-3-031-46661-8_3

the form of sinusoidal functions of different frequencies, with each embedding dimension corresponding to a sinusoid whose wavelengths form a geometric progression. To date, positional embedding has been a routine for Transformer-based models that deal with multivariate time series classification problems.

Despite the widespread use, there have been debates [27,31] around the necessity of positional embedding, and a comprehensive investigation of positional embedding's effectiveness on various Transformer-based models is still to be developed. Firstly, Transformer-based models [14,33] that contain position-sensitive modules (e.g., convolutional and recurrent layers) can automatically learn the position information, making positional embedding redundant to some extent. This point is supported by studies in other fields [18,28] suggesting positional embedding may be unnecessary and replaced with position-sensitive layers. Secondly, positional embedding has inherent limitations that may potentially impair the classifier's performance. Since positional embedding is hand-crafted, it may bring inductive bias that may adversely impact the model's performance in some cases. While positional embedding injects the same position tokens into time series of different classes, it poses additional challenges to the classifier in figuring out the differences between sequences with different class labels.

In this paper, we explore the impact of positional embedding on various Transformer-based models to facilitate researchers and practitioners in making informed decisions on whether to incorporate positional embedding in their models for multivariate time series classification. In a nutshell, we make the following contributions in this paper:

- We comprehensively review existing Transformer-based models that contain position-sensitive layers and summarize them into six types of Transformer-based variants.
- We conduct extensive experiments on 30 multivariate time series classification datasets and evaluate the impact of positional embedding on the vanilla Transformer and Transformer-based variants.
- Our results show that positional embedding positively impacts the performance of the vanilla Transformer while negatively influencing the performance of the Transformer-based variants in multivariate time series classification.

2 Background

2.1 Positional Embedding

Positional embedding was first proposed for Transformer in [23], which uses fixed sine and cosine functions of different frequencies to represent the position information, as described below:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right) \end{aligned} \tag{1}$$

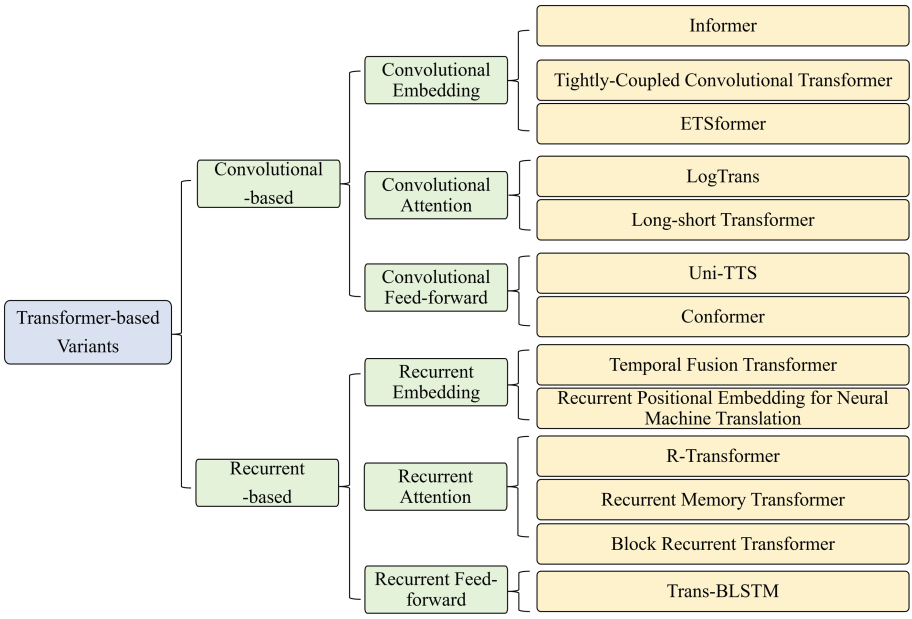

Fig. 1. A summary of Transformer-based variants for modeling sequential data.

where *pos* and i are the position and the dimension indices, respectively. d_{model} is the dimensionality of the input time series, where each dimension of thepositional embedding corresponds to a sinusoid. For any fixed offset k, $PE_{\text{pos}+k}$ is represented as a linear function of PE_{pos}; this enables the model to learn the relative positions easily. The positional embedding is then added to the time series as the input to Transformer.

Considering hand-crafted positional embedding is generally less expressive and adaptive [26], Time Series Transformer (TST) [32] enhances the vanilla Transformer [23] by implementing learnable positional embedding. Specifically, TST shares the same architecture as the vanilla Transformer, which stacks several basic blocks, each consisting of scaled dot-product multi-head attention and a feed-forward network (FFN) to leverage temporal data. But it differs in initializing the positional embedding using fixed values and then updating the embedding jointly with other model parameters through the training procedure.

2.2 Transformer-Based Variants

Current studies often incorporate convolutional or recurrent layers in the vanilla Transformer architecture in dealing with sequence-related tasks, including time series analysis. Figure 1 summarize these Transformer-based variants into six categories of representative methods, as detailed below.

– **Convolutional Embedding**: Methods in this category, namely Informer [33], Tightly-Coupled Convolutional Transformer (TCCT) [21], and ETS-

former [27], implement a convolutional layer to obtain convolutional embeddings, which map the raw input sequences to a latent space before feeding them to the transformer block.
- **Convolutional Attention**: Instead of calculating the point-wise attention, LogTrans [13] and Long-short Transformer [34] use the convolutional layer to calculate the attention matrix (including queries, keys, and values) of segments to leverage the local temporal information.
- **Convolutional Feed-forward**: Uni-TTS [17] and Conformer [8] implement a convolutional layer after the multi-head attention as the feed-forward layer (or part of the feed-forward layer) to capture local temporal correlations.
- **Recurrent Embedding**: Temporal Fusion Transformer (TFT) [14] and the work in [3] use a recurrent layer to encode content-based order dependencies into the input sequence.
- **Recurrent Attention**: Recurrent Memory Transformer [2], Block Recurrent Transformer [11], and R-Transformer [25] use a recurrent neural net to calculate the attention matrix, which harnesses the temporal information more effectively when compared with the point-wise attention.
- **Recurrent Feed-forward**: Instead of point-wise feed-forward, TRANS-BLSTM [10] uses a recurrent layer after multi-head attention to harness non-linear temporal dependencies.

3 Problem Definition

A multivariate time series sequence can be described as: $X = \{x_1, x_2, \ldots x_T\}$, where $x_i \in \mathbb{R}^N$, $i \in \{1, 2, \cdots, T\}$, T is the maximum length of the sequence, and N is the number of variables. A dataset contains multiple (sequence, label) pairs and is denoted by $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$, where each y_k denotes a label, $k \in \{1, 2, \cdots, n\}$. The objective of multivariate time series classification is to train a classifier to map the input sequences to probability distributions over the classes for the dataset D.

4 Methodology

We call TST [32] the **basic model**. To avoid having to compare all the related studies exhaustively, we design six Transformer-based variants based on the six types of techniques that are incorporated in the related studies (shown in Fig. 1), respectively. We further identify three configurable components of a Transformer architecture (shown in Fig. 2) as the input embedding layer (which projects the input time series into the latent space), the projection layer (which calculates the attention matrix), and the feed-forward layer (which leverages non-linear relationships). For each variant, we try different techniques (e.g., a Linear layer, a Convolutional layer, or a Gated Recurrent Unit) in each layer/component, as detailed in Table 1.

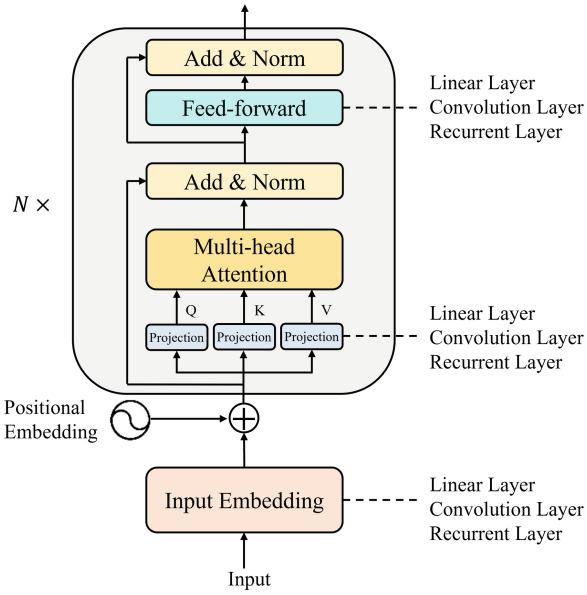

Fig. 2. A general architecture of transformer-based variants for modeling sequential data. The corresponding relations between configurable components and the respective candidate techniques are indicated by dashed lines.

4.1 Basic Model

The **basic model** adopts linear layers in all three components. In this case, for each sample $\mathbf{x_t} \in \mathbb{R}^M : \mathbf{X} \in \mathbb{R}^{M \times T} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x_T}]$, where T is the sequence length and M is the variable number. The input embedding can be described as:

$$U_t = W^x x_t + b^x \tag{2}$$

where $t = 0, 1, ..., T$ is the time stamp index, $W^x \in \mathbb{R}^{M \times d_k}$ and $b^x \in \mathbb{R}^{d_k}$ are learnable parameters. The projection layer can be described as:

$$\begin{aligned} Q &= W^Q U_t + b^Q \\ K &= W^K U_t + b^K \\ V &= W^V U_t + b^V \end{aligned} \tag{3}$$

where $W^Q \in \mathbb{R}^{d_k \times d_k}$, $W^K \in \mathbb{R}^{d_k \times d_k}$, $W^V \in \mathbb{R}^{d_k \times d_k}$, $b^Q \in \mathbb{R}^{d_k}$, $b^K \in \mathbb{R}^{d_k}$, and $b^V \in \mathbb{R}^{d_k}$ are are learnable parameters. We use standard scaled Dot-Product attention proposed in the vanilla Transformer [23] for self-attention calculation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V. \tag{4}$$

The feed-forward layer can be described as:

$$FFN(x) = \text{ReLU}\left(W_1 x + b_1\right) W_2 + b_2 \tag{5}$$

Table 1. Configurations for the basic model and six variants. *ConvEmbedding* means Convolutional Embedding Variant; *RecEmbedding* means Recurrent Embedding Variant; the same naming rule applies to other models.

Model	Input Embedding	Projection	Feed-forward
Basic Model	Linear	Linear	Linear
ConvEmbedding	Convolutional Layer	Linear	Linear
ConvAttention	Linear	Convolutional Layer	Linear
ConvFFD	Linear	Linear	Convolutional Layer
RecEmbedding	Gated Recurrent Unit	Linear	Linear
RecAttention	Linear	Gated Recurrent Unit	Linear
RecFFD	Linear	Linear	Gated Recurrent Unit

where $W_1 \in \mathbb{R}^{d_k \times d_k}$, $W_2 \in \mathbb{R}^{d_k \times d_k}$, $b_1 \in \mathbb{R}^{d_k}$, and $b_2 \in \mathbb{R}^{d_k}$ are all leanable parameters.

4.2 Convolutional-Based Variants

We refer to the architectures that employ convolutional layers in any of the three components (input embedding layer, projection layer, or feed-forward layer) as convolutional-based variants. Here, we utilize a one-dimensional convolutional layer with a kernel size of 3. We also set the padding to 1 to preserve the lengths of representations. In the following, we illustrate our convolutional-based variants one by one.

Convolutional Embedding Variant replaces the linear layer with the convolution layer in the input embedding layer, which is formulated below:

$$U_t = W^x * x_t + b^x \tag{6}$$

where $*$ is the convolutional operation, $W^x \in \mathbb{R}^{M \times d_k \times P}$ and $b^x \in \mathbb{R}^M$ are learnable parameters, and P is the kernel size.

Convolutional Attention Variant replaces the linear layer with the convolution layer in the projection layer, which is formulated below:

$$\begin{aligned} Q &= W^Q * U_t + b^Q \\ K &= W^K * U_t + b^K \\ V &= W^V * U_t + b^V \end{aligned} \tag{7}$$

where $W^Q \in \mathbb{R}^{d_k \times d_k \times P}$, $W^K \in \mathbb{R}^{d_k \times d_k \times P}$, $W^V \in \mathbb{R}^{d_k \times d_k \times P}$, $b^Q \in \mathbb{R}^{d_k}$, $b^K \in \mathbb{R}^{d_k}$, and $b^V \in \mathbb{R}^{d_k}$ are learnable parameters.

Convolutional Feed-forward Variant formulated the linear layer with the convolution layer in the feed-forward layer, which is described below:

$$FFN(x) = \mathrm{ReLU}\left(W_1 * x + b_1\right) * W_2 + b_2 \tag{8}$$

where $W_1 \in \mathbb{R}^{d_k \times d_k \times P}$, $W_2 \in \mathbb{R}^{d_k \times d_k \times P}$, $b_1 \in \mathbb{R}^{d_k}$, and $b_2 \in \mathbb{R}^{d_k}$ are the leanable parameters.

4.3 Recurrent-Based Variants

We name the architectures that use recurrent layers in any of the three components (input embedding layer, projection layer, or feed-forward layer) as recurrent-based variants. Here, we use Gate Recurrent Unit (GRU) [5] as the recurrent layer. In the following, we illustrate our recurrent-based Variants one by one.

Recurrent Embedding Variant replaces the linear layer with the GRU in the input embedding layer, which is formulated below:

$$\begin{aligned} r_t &= \sigma\left(W^x_{ir}x_t + b^x_{ir} + W^x_{hr}U_{(t-1)} + b^x_{hr}\right) \\ z_t &= \sigma\left(W^x_{iz}x_t + b^x_{iz} + W^x_{hz}h_{(t-1)} + b^x_{hz}\right) \\ n_t &= \tanh\left(W^x_{in}x_t + b^x_{in} + r_t \circ \left(W^x_{hn}h_{(t-1)} + b^x_{hn}\right)\right) \\ h_t &= (1 - z_t) \circ n_t + z_t \circ h_{(t-1)} \\ U_t &= Concat(h_1, h_2, ..., h_T) \end{aligned} \tag{9}$$

where $W^x_{ir} \in \mathbb{R}^{M\times d_k}$, $W^x_{iz} \in \mathbb{R}^{M\times d_k}$, $W^x_{in} \in \mathbb{R}^{M\times d_k}$, $W^x_{hr} \in \mathbb{R}^{d_k\times d_k}$, $W^x_{hz} \in \mathbb{R}^{d_k\times d_k}$, $W^x_{hn} \in \mathbb{R}^{d_k\times d_k}$, $b^x_{ir} \in \mathbb{R}^{d_k}$, $b^x_{hr} \in \mathbb{R}^{d_k}$, $b^x_{iz} \in \mathbb{R}^{d_k}$, $b^x_{hz} \in \mathbb{R}^{d_k}$, $b^x_{in} \in \mathbb{R}^{d_k}$, and $b^x_{hn} \in \mathbb{R}^{d_k}$ are learnable parameters, $\circ$ is the Hadamard product.

Recurrent Attention Variant replaces the linear layer with the GRU in the projection layer. Since the calculation processes of all the matrices are similar, for simplicity, we only present the calculation process of the query matrix Q in the projection layer below:

$$\begin{aligned} r_t &= \sigma\left(W^Q_{ir}U_t + b^Q_{ir} + W^Q_{hr}U_{(t-1)} + b^Q_{hr}\right) \\ z_t &= \sigma\left(W^Q_{iz}U_t + b^Q_{iz} + W^Q_{hz}h_{(t-1)} + b^Q_{hz}\right) \\ n_t &= \tanh\left(W^Q_{in}U_t + b^Q_{in} + r_t \circ \left(W^Q_{hn}h_{(t-1)} + b^Q_{hn}\right)\right) \\ h_t &= (1 - z_t) \circ n_t + z_t \circ h_{(t-1)} \\ Q &= Concat(h_1, h_2, ..., h_T) \end{aligned} \tag{10}$$

where $W^Q_{ir} \in \mathbb{R}^{d_k\times d_k}$, $W^Q_{iz} \in \mathbb{R}^{d_k\times d_k}$, $W^Q_{in} \in \mathbb{R}^{d_k\times d_k}$, $W^Q_{hr} \in \mathbb{R}^{d_k\times d_k}$, $W^Q_{hz} \in \mathbb{R}^{d_k\times d_k}$, $W^Q_{hn} \in \mathbb{R}^{d_k\times d_k}$, $b^Q_{ir} \in \mathbb{R}^{d_k}$, $b^Q_{hr} \in \mathbb{R}^{d_k}$, $b^Q_{iz} \in \mathbb{R}^{d_k}$, $b^Q_{hz} \in \mathbb{R}^{d_k}$, $b^Q_{in} \in \mathbb{R}^{d_k}$, and $b^Q_{hn} \in \mathbb{R}^{d_k}$ are learnable parameters.

Recurrent Feed-forward Variant replaces the linear layer with the GRU in the feed-forward layer, which is formulated below:

$$\begin{aligned} r_t &= \sigma\left(W_{ir}U_t + b_{ir} + W_{hr}U_{(t-1)} + b_{hr}\right) \\ z_t &= \sigma\left(W_{iz}U_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}\right) \\ n_t &= \tanh\left(W_{in}U_t + b_{in} + r_t \circ \left(W_{hn}h_{(t-1)} + b_{hn}\right)\right) \\ h_t &= (1 - z_t) \circ n_t + z_t \circ h_{(t-1)} \\ O &= Concat(h_1, h_2, ..., h_T) \end{aligned} \tag{11}$$

where $W_{ir} \in \mathbb{R}^{d_k \times d_k}$, $W_{iz} \in \mathbb{R}^{d_k \times d_k}$, $W_{in} \in \mathbb{R}^{d_k \times d_k}$, $W_{hr} \in \mathbb{R}^{d_k \times d_k}$, $W_{hz} \in \mathbb{R}^{d_k \times d_k}$, $W_{hn} \in \mathbb{R}^{d_k \times d_k}$, $b_{ir} \in \mathbb{R}^{d_k}$, $b_{hr} \in \mathbb{R}^{d_k}$, $b_{iz} \in \mathbb{R}^{d_k}$, $b_{hz} \in \mathbb{R}^{d_k}$, $b_{in} \in \mathbb{R}^{d_k}$, and $b_{hn} \in \mathbb{R}^{d_k}$ are learnable parameters, and O is the final output of the feed-forward layer.

5 Experiments

We empirically evaluate the impact of positional embedding on the performance of the basic model and transformer-based variants (illustrated in Sect. 4) for multivariate time series classification. We report our experimental configurations and discuss the results in the following subsections.

5.1 Datasets

We selected 30 public multivariate time series datasets from the UEA Time Series Classification Repository [7]. All datasets were pre-split into training and test sets[1]. We normalized all datasets to zero mean and unit standard deviation and applied zero padding to ensure all the sequences in each dataset bear the same length.

5.2 Model Configuration and Evaluation Metrics

We trained the basic model and six variants for 500 epochs using the Adam optimizer [12] on all the datasets with and without the learnable positional embedding. Besides, we applied an adaptive learning rate, which was reduced by a factor of 10 after every 100 epochs, and employed dropout regularization to prevent overfitting. Table 2 summarizes our model configurations for each dataset.

We evaluate the models using two metrics: *accuracy* and macro *F1-Score*. To mitigate the effect of randomized parameter initialization, we repeated the training and test procedures five times and took the average as the final results.

5.3 Results and Analysis

Table 3 and Table 4 show the methods' performance on the 30 datasets. The results show positional embedding positively impacts the basic model—with positional embedding, the basic model's performance improves by 17.5% and 14.3% in accuracy and macro F1-Score, respectively. This reveals the significance of enabling the basic model to leverage the position information (e.g., via positional embedding) in solving the multivariate time series classification problem.

In contrast, positional embedding negatively impacts the performance of the Transformer-based variants. Without positional embedding, convolutional

[1] Details about datasets and train-test split can be found at http://www.timeseriesclassification.com/dataset.php.

Table 2. Model configurations.

Dataset	Learning Rate	#Layer	Batch Size	Dropout	#Attention head
ArticularyWordRecognition	0.01	3	32	0.01	2
AtrialFibrillation	0.01	2	16	0.01	2
BasicMotions	0.00001	2	16	0.01	2
CharacterTrajectories	0.01	2	16	0.01	2
Cricket	0.01	2	16	0.01	2
DuckDuckGeese	0.001	4	8	0.3	5
EigenWorms	0.01	1	1	0.01	2
Epilepsy	0.00001	4	16	0.01	2
EthanolConcentration	0.001	2	16	0.01	4
ERing	0.00001	2	16	0.01	2
FaceDetection	0.00001	2	16	0.01	2
FingerMovements	0.001	2	16	0.01	2
HandMovementDirection	0.01	2	16	0.1	2
Handwriting	0.01	5	16	0.01	2
Heartbeat	0.00001	2	16	0.01	2
JapaneseVowels	0.01	3	16	0.3	2
Libras	0.01	5	16	0.01	2
LSST	0.01	2	16	0.01	2
MotorImagery	0.00001	2	16	0.01	2
NATOPS	0.00001	3	16	0.01	2
PenDigits	0.001	2	16	0.01	2
PEMS-SF	0.00001	2	16	0.01	2
Phoneme	0.00001	3	16	0.01	2
RacketSports	0.00001	2	16	0.1	4
SelfRegulationSCP1	0.00001	3	16	0.1	2
SelfRegulationSCP2	0.00001	2	16	0.01	2
SpokenArabicDigits	0.00001	3	16	0.1	2
StandWalkJump	0.01	3	16	0.01	2
UWaveGestureLibrary	0.01	2	16	0.01	2

embedding (i.e., ConvEmbedding in Table 1) and recurrent embedding (i.e., RecEmbedding in Table 1) models outperformed all other variants, achieving the best accuracy of 56.21% and 56.17%, respectively, and the best macro F1-Scores of 0.528 and 0.5375, respectively. These two models differ from all other models in that their input embedding layers encode the position information when projecting the raw data to a latent space, making the position information accessible by subsequent layers for feature extraction and resulting in superior performance. Incorporating positional embedding decreased the average accuracy of the variants by 12.7% (convolutional embedding), 9.1% (convolutional attention), 18.6% (convolutional feed-forward), 22.1% (recurrent embedding),

Table 3. Accuracy of different models on 30 benchmark datasets.

	ArticularyWordRecognition	AtrialFibrillation	BasicMotions	CharacterTrajectories	Cricket	DuckDuckGeese
BasicModel (w/ PE)	0.4788	0.4524	1.0000	0.5140	0.9643	0.7667
BasicModel (w/o PE)	0.1280	0.2949	1.0000	0.4684	0.9395	0.5847
ConvEmbedding (w/ PE)	0.5125	0.4333	1.0000	0.4560	0.7468	0.6922
ConvEmbedding (w/o PE)	0.6774	0.7037	1.0000	0.6207	0.8016	0.7145
ConvAttention (w/ PE)	0.5413	0.5238	1.0000	0.2120	0.6515	0.6257
ConvAttention (w/o PE)	0.5091	0.4000	1.0000	0.1975	0.7433	0.3902
ConvFFD (w/ PE)	0.5232	0.4524	1.0000	0.2834	0.8775	0.5858
ConvFFD (w/o PE)	0.6483	0.5238	1.0000	0.3000	0.9623	0.6472
RecEmbedding (w/ PE)	0.5580	0.4524	1.0000	0.4227	0.8442	0.4800
RecEmbedding (w/o PE)	0.7321	0.6111	1.0000	0.5478	0.9339	0.6608
RecAttention (w/ PE)	0.7312	0.6444	1.0000	0.3842	0.6938	0.3120
RecAttention (w/o PE)	0.7548	0.6768	1.0000	0.7450	0.8371	0.6444
RecFFD (w/ PE)	0.6052	0.4167	1.0000	0.3405	0.6660	0.6065
RecFFD (w/o PE)	0.6890	0.5500	1.0000	0.5001	0.7978	0.6419
	EigenWorms	Epilepsy	JapaneseVowels	Libras	LSST	MotorImagery
BasicModel (w/ PE)	0.4447	0.8153	0.9616	0.0113	0.1716	0.5801
BasicModel (w/o PE)	0.4186	0.7753	0.9346	0.1527	0.1770	0.4332
ConvEmbedding (w/ PE)	0.3556	0.8156	0.9526	0.0317	0.1719	0.4264
ConvEmbedding (w/o PE)	0.4843	0.8725	0.9633	0.0760	0.1086	0.7525
ConvAttention (w/ PE)	0.3792	0.7377	0.7418	0.0878	0.1365	0.5000
ConvAttention (w/o PE)	0.3844	0.5564	0.7491	0.1352	0.1845	0.6420
ConvFFD (w/ PE)	0.4716	0.8041	0.9688	0.0298	0.0987	0.4391
ConvFFD (w/o PE)	0.4872	0.8954	0.9721	0.0490	0.1269	0.4585
RecEmbedding (w/ PE)	0.6724	0.8037	0.9566	0.0691	0.0461	0.5525
RecEmbedding (w/o PE)	0.7053	0.8538	0.9669	0.1070	0.0943	0.6326
RecAttention (w/ PE)	0.4171	0.6734	0.9668	0.1023	0.1006	0.6302
RecAttention (w/o PE)	0.6420	0.8163	0.9608	0.1897	0.0904	0.7551
RecFFD (w/ PE)	0.4199	0.7711	0.9702	0.1014	0.1370	0.4688
RecFFD (w/o PE)	0.6545	0.8353	0.9709	0.1159	0.1664	0.4719
	NATOPS	PenDigits	PEMS-SF	Phoneme	EthanolConcentration	ERing
BasicModel (w/ PE)	0.2834	0.1639	0.6204	0.0164	0.3880	0.6659
BasicModel (w/o PE)	0.1198	0.1203	0.8091	0.0110	0.0627	0.6065
ConvEmbedding (w/ PE)	0.2249	0.6416	0.8076	0.0126	0.1086	0.6982
ConvEmbedding (w/o PE)	0.3074	0.6452	0.8770	0.0291	0.1979	0.7695
ConvAttention (w/ PE)	0.2111	0.0784	0.8717	0.0170	0.0632	0.6660
ConvAttention (w/o PE)	0.2699	0.5021	0.8953	0.0211	0.1254	0.6913
ConvFFD (w/ PE)	0.2339	0.2036	0.8155	0.0142	0.0627	0.4966
ConvFFD (w/o PE)	0.2796	0.3388	0.8816	0.0142	0.1340	0.7357
RecEmbedding (w/ PE)	0.2951	0.3561	0.6314	0.0059	0.0618	0.4318
RecEmbedding (w/o PE)	0.4215	0.4066	0.8277	0.0106	0.4755	0.8008
RecAttention (w/ PE)	0.2638	0.1491	0.5946	0.0111	0.0985	0.6291
RecAttention (w/o PE)	0.4958	0.1711	0.8348	0.0243	0.1236	0.6488
RecFFD (w/ PE)	0.2797	0.1003	0.5953	0.0074	0.0618	0.7222
RecFFD (w/o PE)	0.4231	0.2755	0.8333	0.0123	0.1254	0.7735
	FaceDetection	FingerMovements	HandMovementDirection	Handwriting	Heartbeat	RacketSports
BasicModel (w/ PE)	0.6285	0.6075	0.2691	0.0239	0.7528	0.0879
BasicModel (w/o PE)	0.5487	0.3925	0.1830	0.0460	0.8627	0.0546
ConvEmbedding (w/ PE)	0.6461	0.5405	0.3032	0.0266	0.8663	0.4439
ConvEmbedding (w/o PE)	0.5589	0.5861	0.3224	0.0307	0.7506	0.4530
ConvAttention (w/ PE)	0.6762	0.7082	0.2908	0.0154	0.7238	0.2111
ConvAttention (w/o PE)	0.6816	0.7552	0.3442	0.0664	0.8663	0.5574
ConvFFD (w/ PE)	0.6457	0.6725	0.2858	0.0364	0.7258	0.1691
ConvFFD (w/o PE)	0.6347	0.7242	0.2950	0.4540	0.7272	0.3109
RecEmbedding (w/ PE)	0.6793	0.5187	0.4222	0.0400	0.3610	0.1716
RecEmbedding (w/o PE)	0.5500	0.5233	0.3779	0.0570	0.7096	0.2121
RecAttention (w/ PE)	0.6797	0.5028	0.2083	0.0318	0.3610	0.1970
RecAttention (w/o PE)	0.5425	0.6731	0.2559	0.0513	0.7528	0.3110
RecFFD (w/ PE)	0.6648	0.2550	0.4335	0.0189	0.7238	0.1253
RecFFD (w/o PE)	0.5527	0.4167	0.4454	0.0352	0.7435	0.2003
	SelfRegulationSCP1	SelfRegulationSCP2	SpokenArabicDigits	StandWalkJump	UWaveGestureLibrary	Average
BasicModel (w/ PE)	0.8494	0.5259	0.1111	0.5694	0.5284	0.4915
BasicModel (w/o PE)	0.8176	0.5125	0.1111	0.2778	0.2865	0.4183
ConvEmbedding (w/ PE)	0.8720	0.5847	0.4433	0.2778	0.3656	0.4986
ConvEmbedding (w/o PE)	0.8846	0.5694	0.5988	0.5500	0.3938	0.5621
ConvAttention (w/ PE)	0.8188	0.5207	0.4681	0.4524	0.4763	0.4623
ConvAttention (w/o PE)	0.8592	0.5000	0.6620	0.5500	0.3818	0.5042
ConvFFD (w/ PE)	0.8375	0.5250	0.5591	0.1944	0.3472	0.4607
ConvFFD (w/o PE)	0.8979	0.6080	0.6418	0.5500	0.5496	0.5465
RecEmbedding (w/ PE)	0.7935	0.2500	0.5833	0.4778	0.4031	0.4600
RecEmbedding (w/o PE)	0.8472	0.2500	0.7232	0.8182	0.4338	0.5617
RecAttention (w/ PE)	0.8494	0.5000	0.5112	0.5500	0.4113	0.4553
RecAttention (w/o PE)	0.8746	0.5710	0.5420	0.7167	0.3374	0.5531
RecFFD (w/ PE)	0.8658	0.4486	0.4491	0.4667	0.3620	0.4512
RecFFD (w/o PE)	0.8843	0.5752	0.4555	0.4615	0.5370	0.5222

Table 4. Macro F1-Score of different models on 30 benchmark datasets.

	ArticularyWordRecognition	AtrialFibrillation	BasicMotions	CharacterTrajectories	Cricket	DuckDuckGeese
BasicModel (w/ PE)	0.5395	0.3523	1.0000	0.5783	0.9581	0.7177
BasicModel (w/o PE)	0.0988	0.3297	1.0000	0.5687	0.9156	0.4478
ConvEmbedding (w/ PE)	0.6170	0.4139	1.0000	0.5554	0.7208	0.5365
ConvEmbedding (w/o PE)	0.7226	0.6035	1.0000	0.6734	0.8285	0.6379
ConvAttention (w/ PE)	0.5094	0.4000	1.0000	0.1879	0.6382	0.3959
ConvAttention (w/o PE)	0.5821	0.5157	1.0000	0.3141	0.7758	0.4760
ConvFFD (w/ PE)	0.5921	0.3529	1.0000	0.3892	0.8719	0.5365
ConvFFD (w/o PE)	0.7152	0.3333	1.0000	0.4318	0.9582	0.6360
RecEmbedding (w/ PE)	0.6272	0.3591	1.0000	0.4510	0.8194	0.4748
RecEmbedding (w/o PE)	0.7605	0.4577	1.0000	0.6335	0.9025	0.5304
RecAttention (w/ PE)	0.7599	0.4603	1.0000	0.4473	0.7108	0.3354
RecAttention (w/o PE)	0.7767	0.4553	1.0000	0.7868	0.8600	0.5338
RecFFD (w/ PE)	0.6615	0.4142	1.0000	0.4351	0.7121	0.5338
RecFFD (w/o PE)	0.7314	0.5161	1.0000	0.5712	0.8176	0.6114
	EigenWorms	Epilepsy	JapaneseVowels	Libras	LSST	MotorImagery
BasicModel (w/ PE)	0.4020	0.8169	0.9670	0.0321	0.2009	0.4325
BasicModel (w/o PE)	0.3912	0.7667	0.9439	0.1307	0.1297	0.5799
ConvEmbedding (w/ PE)	0.2471	0.7568	0.9514	0.0148	0.1504	0.4265
ConvEmbedding (w/o PE)	0.2849	0.8345	0.9600	0.0403	0.1220	0.4500
ConvAttention (w/ PE)	0.2460	0.5912	0.7222	0.0581	0.0555	0.4120
ConvAttention (w/o PE)	0.3592	0.7100	0.6509	0.0158	0.1317	0.6393
ConvFFD (w/ PE)	0.3695	0.7667	0.9656	0.0568	0.0585	0.4390
ConvFFD (w/o PE)	0.4923	0.8941	0.9744	0.0744	0.0829	0.4623
RecEmbedding (w/ PE)	0.5984	0.7479	0.9566	0.1339	0.0752	0.5498
RecEmbedding (w/o PE)	0.6606	0.8557	0.9679	0.1508	0.0681	0.6291
RecAttention (w/ PE)	0.3231	0.5496	0.9686	0.0690	0.0950	0.4630
RecAttention (w/o PE)	0.5197	0.8072	0.9673	0.2292	0.0830	0.5526
RecFFD (w/ PE)	0.3738	0.7462	0.9721	0.1448	0.1312	0.4533
RecFFD (w/o PE)	0.6519	0.8410	0.9741	0.1512	0.0724	0.4547
	NATOPS	PenDigits	PEMS-SF	Phoneme	EthanolConcentration	ERing
BasicModel (w/ PE)	0.3127	0.0869	0.7693	0.0236	0.2152	0.6486
BasicModel (w/o PE)	0.2291	0.0747	0.6615	0.0191	0.1432	0.5273
ConvEmbedding (w/ PE)	0.3257	0.6173	0.7872	0.0234	0.1739	0.6498
ConvEmbedding (w/o PE)	0.3974	0.6262	0.8655	0.0247	0.2096	0.7679
ConvAttention (w/ PE)	0.2333	0.1291	0.8543	0.0099	0.0777	0.5992
ConvAttention (w/o PE)	0.3031	0.4512	0.8900	0.0259	0.2224	0.6600
ConvFFD (w/ PE)	0.2121	0.2591	0.7932	0.0249	0.1432	0.5523
ConvFFD (w/o PE)	0.3564	0.3129	0.8763	0.0207	0.1581	0.7092
RecEmbedding (w/ PE)	0.3635	0.4070	0.6814	0.0142	0.1419	0.4622
RecEmbedding (w/o PE)	0.3715	0.4176	0.7929	0.0149	0.2600	0.7814
RecAttention (w/ PE)	0.1894	0.2101	0.6375	0.0205	0.1483	0.5415
RecAttention (w/o PE)	0.3423	0.2118	0.8277	0.0234	0.2217	0.6511
RecFFD (w/ PE)	0.2344	0.1380	0.6328	0.0165	0.1419	0.6853
RecFFD (w/o PE)	0.4271	0.3123	0.7916	0.0238	0.2224	0.7615
	FaceDetection	FingerMovements	HandMovementDirection	Handwriting	Heartbeat	RacketSports
BasicModel (w/ PE)	0.6270	0.5580	0.2603	0.0441	0.6119	0.1502
BasicModel (w/o PE)	0.5483	0.4029	0.1593	0.0680	0.6167	0.1304
ConvEmbedding (w/ PE)	0.6454	0.4931	0.2703	0.0549	0.6167	0.2044
ConvEmbedding (w/o PE)	0.5563	0.5310	0.2960	0.0598	0.6119	0.3170
ConvAttention (w/ PE)	0.6814	0.5710	0.3052	0.0369	0.7351	0.2400
ConvAttention (w/o PE)	0.6762	0.5181	0.1970	0.0622	0.5950	0.2582
ConvFFD (w/ PE)	0.6396	0.5612	0.3102	0.0454	0.6144	0.2732
ConvFFD (w/o PE)	0.5544	0.5923	0.2349	0.0631	0.6093	0.2814
RecEmbedding (w/ PE)	0.6793	0.5154	0.2097	0.0713	0.4561	0.1677
RecEmbedding (w/o PE)	0.5499	0.5211	0.3298	0.0795	0.5702	0.2394
RecAttention (w/ PE)	0.6793	0.4997	0.1763	0.0594	0.4561	0.2551
RecAttention (w/o PE)	0.5419	0.5457	0.2305	0.0653	0.6091	0.2499
RecFFD (w/ PE)	0.6642	0.4031	0.3617	0.0465	0.5950	0.1940
RecFFD (w/o PE)	0.5523	0.4174	0.3231	0.0609	0.5986	0.3186
	SelfRegulationSCP1	SelfRegulationSCP2	SpokenArabicDigits	StandWalkJump	UWaveGestureLibrary	Average
BasicModel (w/ PE)	0.8167	0.4669	0.2222	0.4260	0.5309	0.4748
BasicModel (w/o PE)	0.8120	0.5040	0.2222	0.2947	0.3282	0.4153
ConvEmbedding (w/ PE)	0.8632	0.5131	0.5001	0.2871	0.3785	0.4757
ConvEmbedding (w/o PE)	0.8839	0.5448	0.6328	0.4611	0.3677	0.5280
ConvAttention (w/ PE)	0.2548	0.4346	0.3021	0.2353	0.3867	0.3898
ConvAttention (w/o PE)	0.8119	0.4434	0.4141	0.2559	0.4812	0.4633
ConvFFD (w/ PE)	0.8249	0.5217	0.6121	0.2508	0.3023	0.4600
ConvFFD (w/o PE)	0.8976	0.5740	0.6497	0.4496	0.5897	0.5167
RecEmbedding (w/ PE)	0.7074	0.4000	0.5586	0.4680	0.4077	0.4657
RecEmbedding (w/o PE)	0.8132	0.4000	0.6771	0.7370	0.4143	0.5375
RecAttention (w/ PE)	0.8167	0.4511	0.4883	0.4128	0.3663	0.4341
RecAttention (w/o PE)	0.8736	0.5425	0.5469	0.5220	0.3617	0.5151
RecFFD (w/ PE)	0.8524	0.4133	0.5315	0.2353	0.3765	0.4517
RecFFD (w/o PE)	0.8839	0.5533	0.5451	0.4163	0.5811	0.5235

21.5% (recurrent attention), and 15.7% (recurrent feed-forward), respectively. Results for the macro F1-Score show similar trends.

Since the convolutional and recurrent layers can inherently capture the position information from sequential data, it is natural to consider positional embedding redundant for Transformer-based variants. Besides, positional embedding risks introducing inductive bias and contaminating the original data. Specifically, positional embedding injects the same information into sequences of different classes, bringing new challenges to the classifiers; this may also contribute to performance degradation.

Further reflecting on the results, we suggest that positional embedding may not be necessary for Transformer-based variants that already contain position-sensitive modules. In particular, for time series classification tasks, while the classifier focuses on the differences between time series sequences across different classes, positional embedding is content-irrelevant, adding the same position information to all sequences regardless of their class labels. As position-sensitive modules generally consider content information when encoding the position information, redundant content-irrelevant positional embedding may lead the model towards capturing spurious correlations that potentially hinder the classifier's performance.

6 Conclusion

Existing Transformer-based architectures generally contain position-sensitive layers while routinely incorporating positional embedding without comprehensively evaluating its effectiveness on multivariate time series classification. In this paper, we investigate the impact of positional embedding on the vanilla Transformer architecture and six types of Transformer-based variants in multivariate time series classification. Our experimental results on 30 public time series datasets show that positional embedding lifts the performance of the vanilla Transformer while adversely impacting the performance of Transformer-based variants on classification tasks. Our findings refute the necessity of incorporating positional embedding in Transformer-based architectures that already contain position-sensitive layers, such as convolutional or recurrent layers. We also advocate applying position-sensitive layers directly on the input for any Transformer-based architecture that considers using position-sensitive layers to gain better results in multivariate time series classification.

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: a video vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6836–6846 (2021)
2. Bulatov, A., Kuratov, Y., Burtsev, M.S.: Recurrent memory transformer. arXiv preprint arXiv:2207.06881 (2022)

3. Chen, K., Wang, R., Utiyama, M., Sumita, E.: Recurrent positional embedding for neural machine translation. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 1361–1367 (2019)
4. Chen, Z., Chen, D., Zhang, X., Yuan, Z., Cheng, X.: Learning graph structures with transformer for multivariate time series anomaly detection in IoT. IEEE Internet Things J. **9**, 9179–9189 (2021)
5. Cho, K., et al.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
6. Coyle, D., Prasad, G., McGinnity, T.M.: A time-series prediction approach for feature extraction in a brain-computer interface. IEEE Trans. Neural Syst. Rehabil. Eng. **13**(4), 461–467 (2005)
7. Dau, H.A., et al.: Hexagon-ML: the UCR time series classification archive (2018)
8. Gulati, A., et al.: Conformer: convolution-augmented transformer for speech recognition. arXiv preprint arXiv:2005.08100 (2020)
9. Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al.: A survey on vision transformer. IEEE Trans. Pattern Anal. Mach. Intell. **45**(1), 87–110 (2022)
10. Huang, Z., Xu, P., Liang, D., Mishra, A., Xiang, B.: Trans-blstm: transformer with bidirectional LSTM for language understanding. arXiv preprint arXiv:2003.07000 (2020)
11. Hutchins, D., Schlag, I., Wu, Y., Dyer, E., Neyshabur, B.: Block-recurrent transformers. arXiv preprint arXiv:2203.07852 (2022)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Li, S., et al.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Adv. Neural Inf. Process. Syst. **32** (2019)
14. Lim, B., Arık, S.Ö., Loeff, N., Pfister, T.: Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int. J. Forecast. **37**(4), 1748–1764 (2021)
15. Liu, M., Kim, Y.: Classification of heart diseases based on ecg signals using long short-term memory. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 2707–2710. IEEE (2018)
16. Liu, M., et al.: Gated transformer networks for multivariate time series classification. arXiv preprint arXiv:2103.14438 (2021)
17. Liu, Y., et al.: Delightfultts: the microsoft speech synthesis system for blizzard challenge 2021. arXiv preprint arXiv:2110.12612 (2021)
18. Pan, Z., Cai, J., Zhuang, B.: Fast vision transformers with hilo attention. arXiv preprint arXiv:2205.13213 (2022)
19. Raganato, A., Tiedemann, J.: An analysis of encoder representations in transformer-based machine translation. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. The Association for Computational Linguistics (2018)
20. Serdyuk, D., Braga, O., Siohan, O.: Transformer-based video front-ends for audio-visual speech recognition, p. 15. arXiv preprint arXiv:2201.10439 (2022)
21. Shen, L., Wang, Y.: TCCT: tightly-coupled convolutional transformer on time series forecasting. Neurocomputing **480**, 131–145 (2022)
22. Song, Q., Sun, B., Li, S.: Multimodal sparse transformer network for audio-visual speech recognition. IEEE Trans. Neural Netw. Learn. Syst. (2022)
23. Vaswani, A., et al.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017)

24. Wang, Q., et al.: Learning deep transformer models for machine translation. arXiv preprint arXiv:1906.01787 (2019)
25. Wang, Z., Ma, Y., Liu, Z., Tang, J.: R-transformer: recurrent neural network enhanced transformer. arXiv preprint arXiv:1907.05572 (2019)
26. Wen, Q., et al.: Transformers in time series: a survey. arXiv preprint arXiv:2202.07125 (2022)
27. Woo, G., Liu, C., Sahoo, D., Kumar, A., Hoi, S.: Etsformer: exponential smoothing transformers for time-series forecasting. arXiv e-prints arXiv:2202.01381 (2022)
28. Wu, H., et al.: CVT: introducing convolutions to vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 22–31 (2021)
29. Yang, C., Jiang, W., Guo, Z.: Time series data classification based on dual path CNN-RNN cascade network. IEEE Access **7**, 155304–155312 (2019)
30. Yuan, Y., Lin, L.: Self-supervised pretraining of transformers for satellite image time series classification. IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens. **14**, 474–487 (2020)
31. Zeng, A., Chen, M., Zhang, L., Xu, Q.: Are transformers effective for time series forecasting? arXiv preprint arXiv:2205.13504 (2022)
32. Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., Eickhoff, C.: A transformer-based framework for multivariate time series representation learning. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 2114–2124 (2021)
33. Zhou, H., et al.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115 (2021)
34. Zhu, C., et al.: Long-short transformer: efficient transformers for language and vision. Adv. Neural. Inf. Process. Syst. **34**, 17723–17736 (2021)