frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 29 January 2025
DOI 10.3389/fnins.2024.1516868

OPEN ACCESS

EDITED BY
Zhuo Zou,
Fudan University, China

REVIEWED BY
Lei Deng,
Tsinghua University, China
Anguo Zhang,
University of Macau, China

*CORRESPONDENCE
Tielin Zhang
✉ zhangtielin@ion.ac.cn
Bo Xu
✉ xubo@ia.ac.cn

RECEIVED 25 October 2024
ACCEPTED 30 December 2024
PUBLISHED 29 January 2025

CITATION
Wang Q, Zhang D, Cai X, Zhang T and Xu B (2025) Fourier or Wavelet bases as counterpart self-attention in spikformer for efficient visual classification. *Front. Neurosci.* 18:1516868. doi: 10.3389/fnins.2024.1516868

Fourier or Wavelet bases as counterpart self-attention in spikformer for efficient visual classification

Qingyu Wang[1,2], Duzhen Zhang[1], Xinyuan Cai[1], Tielin Zhang[3]* and Bo Xu[1]*

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China, [2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China, [3]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China

Energy-efficient spikformer has been proposed by integrating the biologically plausible spiking neural network (SNN) and artificial transformer, whereby the spiking self-attention (SSA) is used to achieve both higher accuracy and lower computational cost. However, it seems that self-attention is not always necessary, especially in sparse spike-form calculation manners. In this article, we innovatively replace vanilla SSA (using dynamic bases calculating from Query and Key) with spike-form Fourier transform, wavelet transform, and their combinations (using fixed triangular or wavelets bases), based on a key hypothesis that both of them use a set of basis functions for information transformation. Hence, the Fourier-or-Wavelet-based spikformer (FWformer) is proposed and verified in visual classification tasks, including both static image and event-based video datasets. The FWformer can achieve comparable or even higher accuracies (0.4%–1.5%), higher running speed (9%–51% for training and 19%–70% for inference), reduced theoretical energy consumption (20%–25%), and reduced graphic processing unit (GPU) memory usage (4%–26%), compared to the standard spikformer. Our result indicates the continuous refinement of new transformers that are inspired either by biological discovery (spike-form), or information theory (Fourier or Wavelet transform), is promising.

KEYWORDS

spiking neural network, transformer, Fourier/Wavelet transform, visual classification, computational efficiency

1 Introduction

Spiking neural network (SNN) is considered the third generation of artificial neural networks (Maass, 1997) for its biological plausibility of event-driven characteristics. It has also received extensive attention in the computation area of neuromorphic hardware (Davies et al., 2018), exhibiting a remarked lower computational cost on various machine learning tasks, including but not limited to, visual classification (Zhou et al., 2023), temporal auditory recognition (Wang et al., 2023), and reinforcement learning (Tang et al., 2021). The progress in SNN is contributed initially by some key computational modules inspired by the biological brain, for example the receptive-field-like convolutional circuits, self-organized plasticity propagation (Zhang et al., 2021), and other multi-scale inspiration from the single neuron or synapse to the network or cognitive functions. Simultaneously, the SNN also learns from the artificial neural network (ANN) by

borrowing some mathematical optimization algorithms, for example the approximate gradients in backpropagation (BP), various types of loss definitions, and regression configurations.

Even though various advanced architectures have been proposed and contributed ANN to a powerful framework, the efforts to promote its training speed and computational consumption have never been stopped. As the well-known transformer for example, it contains a rich information representation formed by multi-head self-attention, which calculates Query, Key, and Value from the inputs to connect each token in a sequence with every other token. Although having achieved rapid and widespread application, the $\mathcal{O}(N^2)$ complexity (with N representing the sequence length) results in a huge training cost in transformer that can not be neglected. Many works have tried to solve this problem, including but not limited to, replacing self-attention with unparameterized transform formats, for example, using Fourier transform (FNet) (Lee-Thorp et al., 2021) or Gaussian transform (Gaussian attention) (You et al., 2020). Another attempt is to integrate some key features of ANNs and SNNs to exhibit their advantages, such as the higher accuracy performance in ANNs and the lower computational cost in SNNs.

The spikformer (Zhou et al., 2023) explores self-attention in SNN for more advanced deep learning in visual recognition. It introduces a spike-form self-attention called spiking self-attention (SSA). In SSA, the floating Query, Key, and Value signals are sent to leaky-integrated and fire (LIF) neurons to generate spike sequences that only contain binary and sparse 0 and 1 vision information, which results in non-negativeness spiking attention map. This special map doesn't require the complex softmax operation anymore for further normalization, which means a lower computational consumption is needed compared to that in vanilla self-attention. However, even though many efforts have been made, it seems that the SSA still exhibits an $\mathcal{O}(N^2)$ complexity, for which further refinement is necessary. Given binary and sparse spikes for information representation, we here question whether it is still necessary to retain the original complex structure of Self-Attention in spikformer. Here, we give a hypothesis that although self-attention with learning parameters has been generally considered more flexible, it is still not suitable in the spike stream context, since the correlation between sparse spike trains is too weak to form closed similarity. In the field of image processing, Fourier and wavelet transforms have achieved remarkable success in tasks such as image denoising (Tian et al., 2023), edge detection (You et al., 2023), and image compression (Zhang et al., 2020). Fourier transform specializes in global frequency analysis, while Wavelet transform adds multi-resolution capabilities for both global and local feature capture. These techniques not only offer indispensable tools for feature extraction, enabling precise and efficient analysis, but also form a solid theoretical foundation for deep learning models. Hence, an intuitive approach is to convert these sparse spike trains in spatial domains to the equivalent frequency domains with the help of Fourier or wavelet transformation.

Here we propose a new hypothesis: Just like the Fourier transform, self-attention can also be thought of as using a set of basis frequency functions for information representation. The main difference between these two methods is that the Fourier transform uses fixed triangular basis functions to transform signals into the frequency domain, while on the contrary, the self-attention calculates higher-order signal representation from compositions of the input to produce more complex basis functions (*Query* $\times$ *Key*). This understanding may explain why FNet (Lee-Thorp et al., 2021) performs well, since fixed basis functions may also work in some cases by offering structured prior information. Following this perspective, an intuitive plan is to integrate all these key features together, toward a reduced computational cost and accelerated running speed, including unparameterized transforms (e.g., Fourier transform and wavelet transform), and spike-form sparse representation. Our main contributions can be summarized as follows:

- We propose a key hypothesis that the self-attention in transformer works by using a set of basis functions to transform information from Query, Key, and Value sequences, which is very similar to the Fourier transform. Hence, after jointly considering the shortcomings of spikformer, we replaced SSA with spike-form Fourier transform and wavelet transform. Mathematical analysis indicates a reduced time complexity from $\mathcal{O}(Nd^2)$ or $\mathcal{O}(N^2d)$, to $\mathcal{O}(N\log N)$ or $\mathcal{O}(D\log D)+\mathcal{O}(N\log N)$, under the same accuracy performance.
- The results validate that our method achieves superior accuracy on event-based video datasets (improved by 0.3%–1.2%) and comparable performance on spatial image datasets, compared to spikformer with SSA. Furthermore, it exhibits significantly enhanced computational efficiency, reducing memory usage by 4%–26%, reducing theoretical energy consumption by 20%–25%, and achieving ∼9%–51% and 19%–70% improvements in training and inference speeds, respectively.
- We further analyze the orthogonality of self-attention as a set of basis functions. We find during training that the orthogonality is continuously decreasing, which inspires us to use combined different wavelet bases with non-linear, learnable parameters as coefficients to form structured non-orthogonal basis functions. In the second round of experiments, the experiments show even better accuracy performance on event-based video datasets (improved by 0.4%–1.5% compared to spikformer).

2 Related studies

2.1 Vision transformers

The vanilla transformer architecture, initially designed for natural language processing (Vaswani et al., 2017), has demonstrated remarkable success in various other computer-vision tasks, including image classification (Dosovitskiy et al., 2020), semantic segmentation (Wang et al., 2021), object detection (Carion et al., 2020), and low-level image processing (Chen et al., 2021). The critical component that contributes to the success of the transformer is the self-attention mechanism. In Vision transformer (ViT), self-attention can capture global dependencies

between image patches and generate meaningful representations by weighting the features of these patches, using the dot-product operation between Query and Key, followed by the softmax normalization (Katharopoulos et al., 2020). The structure of ViT also fits for conventional SNNs, offering potential transformer-type architectures for achieving higher accuracy performance.

2.2 Spiking neural networks

In contrast to traditional ANNs that employ continuous floating-point values to convey information, SNNs utilize discrete spike sequences for communication, offering a promising energy-efficient and biologically plausible alternative for computation. The critical components of SNNs encompass spiking neuron models, optimization algorithms, and network architectures. Spiking neurons serve as the fundamental non-linear spatial and temporal information processing units in SNNs, responsible for receiving from continuous inputs and converting them to spike sequences. Leaky Integrate-and-Fire (LIF) (Dayan and Abbott, 2005), PLIF (Fang et al., 2021a), Izhikevich (Izhikevich et al., 2004) neurons are commonly used dynamic neuron models in SNNs for their efficiency and simplicity. There are primarily two optimization algorithms employed in deep SNNs: ANN-to-SNN conversion and direct training. In ANN-to-SNN conversion (Rueckauer et al., 2017), a high-performance pre-trained ANN is converted into an SNN by replacing rectified linear unit (ReLU) activation functions with spiking neurons. However, the converted SNN requires significant time steps to accurately approximate the ReLU activation, leading to substantial latency (Han et al., 2020). In direct training, SNNs are unfolded over discrete simulation time steps and trained using backpropagation through time (Shrestha and Orchard, 2018). Since the event-triggered mechanism in spiking neurons is non-differentiable, surrogate gradients are employed to approximate the non-differentiable parts during backpropagation by using some predefined gradient values to replace infinite gradients (Lee et al., 2020).

With the advancements in ANNs, SNNs have improved their performance by incorporating advanced architectures from ANNs. These architectures include spiking recurrent neural networks (Lotfi Rezaabad and Vishwanath, 2020), ResNet-like SNNs (Hu et al., 2021), and spiking graph neural networks (Xu et al., 2021). Recently, exploring transformer in the context of SNNs has received a lot of attention. For example, temporal attention has been proposed to reduce redundant simulation time steps (Yao et al., 2021). Additionally, an ANN-SNN conversion transformer has been introduced, but it still retains vanilla self-attention that does not align with the inherent properties of SNNs (Mueller et al., 2021). Furthermore, spikformer (Zhou et al., 2023) investigates the feasibility of implementing self-attention and transformer in SNNs using a direct training manner.

In this article, we argue that the artificial transformer can be well integrated into SNNs for higher performance, while at the same time, the utilization of SSA in spiking transformer (spikformer) can be further replaced by a special module based on Fourier transform or wavelet transform, which to some extent, indicating an alternative more efficient effort to achieve fast, efficient computation without affecting the accuracy.

3 Background

3.1 Spiking neuron model

The spiking neuron serves as the fundamental unit in SNNs. It receives the current sequence and accumulates membrane potential, which is subsequently compared to a threshold to determine whether a spike should be generated. In this article, we consistently employ LIF at all spiking neuron layers.

The dynamic model of the LIF neuron is described as follows:

$$H[t] = V[t-1] + \frac{1}{\tau}\left(C[t] - (V[t-1] - V_{\text{reset}})\right), \tag{1}$$

$$S[t] = \mathcal{G}\left(H[t] - V_{th}\right), \tag{2}$$

$$V[t] = H[t](1 - S[t]) + V_{\text{reset}}\, S[t], \tag{3}$$

where τ represents the membrane time constant, and $C[t]$ denotes the input current at time step t. When the membrane potential $H[t]$ exceeds the firing threshold V_{th}, the spiking neuron generates a spike $S[t]$. The Heaviside step function $\mathcal{G}(v)$ is defined as 1 when $v \geq 0$ and 0 otherwise. The membrane potential $V[t]$ will transition to the reset potential V_{reset} if there is a spike event, or otherwise it remains unchanged as $H[t]$.

3.2 Spiking self-attention

The spikformer utilizes the SSA as its primary module for extracting sparse visual features and mixing spike sequences. Given input spike sequences denoted as $\boldsymbol{X} \in \mathbb{R}^{T\times N\times D}$, where T, N, and D represent the time steps, sequence length, and feature dimension, respectively, SSA incorporates three key components: Query ($\boldsymbol{Q}$), Key ($\boldsymbol{K}$), and Value ($\boldsymbol{V}$). These components are initially obtained by applying learnable matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{D\times D}$ to $\boldsymbol{X}$. Subsequently, they are transformed into spike sequences through spiking neuron layers, formulated as:

$$\boldsymbol{Q} = \mathcal{SN}(\text{BN}(\boldsymbol{X}\boldsymbol{W}_Q)), \boldsymbol{K} = \mathcal{SN}(\text{BN}(\boldsymbol{X}\boldsymbol{W}_K)), \boldsymbol{V} = \mathcal{SN}(\text{BN}(\boldsymbol{X}\boldsymbol{W}_V)), \tag{4}$$

where $\mathcal{SN}$ denotes the Spiking Neuron Layer, BN denotes batch normalization and $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{T\times N\times D}$. Inspired by vanilla self-attention (Vaswani et al., 2017), SSA adds a scaling factor s to control the large value of the matrix multiplication result, defined as:

$$\begin{aligned} \text{SSA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \mathcal{SN}\left(\boldsymbol{Q}\,\boldsymbol{K}^{\text{T}}\,\boldsymbol{V} * s\right), \\ \boldsymbol{X}' &= \mathcal{SN}(\text{BN}(\text{Dense}(\text{SSA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})))), \end{aligned} \tag{5}$$

where $\boldsymbol{X}' \in \mathbb{R}^{T\times N\times D}$ are the updated spike sequences. It should be noted that SSA operates independently at each time step. In practice, T represents an independent dimension for the $\mathcal{SN}$ layer. In other layers, it is merged with the batch size. Based on Equation 4, the spike sequences $\boldsymbol{Q}$ and $\boldsymbol{K}$ produced by the $\mathcal{SN}$ layers $\mathcal{SN}_Q$ and $\mathcal{SN}_K$, respectively, naturally have non-negative values (0 or 1). Consequently, the resulting attention map is also non-negative. Therefore, according to Equation 5, there is no need for softmax normalization to ensure the non-negativity of the

attention map, and direct multiplication of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ can be performed. This approach significantly improves computational efficiency compared to vanilla self-attention.

However, it is essential to note that SSA remains an operation with a computational complexity of $\mathcal{O}(N^2)$. Although SSA can be decomposed with an $\mathcal{O}(N)$ attention scaling, this complexity hides large constants, causing limited scalability in practical applications. For a more detailed analysis, refer to *time complexity analysis of FW vs. SSA* section. Within the spike-form frameworks, we are firmly of the view that SSA is not essential, and there exist simpler sequence mixing mechanisms that can efficiently extract sparse visual features as alternatives.

3.3 Fourier transform

The Fourier transform (FT) decomposes a function into its constituent frequencies. For the input spike features $\boldsymbol{x} \in \mathbb{R}^{N\times D}$ at a specific time step in $\boldsymbol{X}$, we utilize the FT to transform information from different dimensions, including 1D-FT and 2D-FT.

The discrete 1D-FT along the sequence dimension of $\boldsymbol{x} \in \mathbb{R}^{N\times D}$ to extract sparse visual features is defined by function $\mathcal{F}_{\text{seq}}$:

$$\boldsymbol{x}'_n = \mathcal{F}_{\text{seq}}(\boldsymbol{x}_n) = \sum_{k=0}^{N-1} \boldsymbol{x}_k e^{-\frac{2\pi i}{N}kn}, n = 0, ..., N-1, \tag{6}$$

where i represents the imaginary unit and k represents the frequency index. For each value of n from 0 to $N-1$, the discrete 1D-FT generates a new representation $\boldsymbol{x}'_n \in \mathbb{R}^D$ as a sum of all the original input spike features $\boldsymbol{x}_n \in \mathbb{R}^D$. It is important to note that the weights in Equation 6 are fixed constant and can be pre-calculated for all spike sequences.

Similarly, the discrete 2D-FT along the feature and sequence dimensions is defined by function $\mathcal{F}_{\text{seq}}(\mathcal{F}_{\text{f}})$:

$$\boldsymbol{x}'_n = \mathcal{F}_{\text{seq}}(\mathcal{F}_{\text{f}}(\boldsymbol{x}_n)), n = 0, ..., N-1. \tag{7}$$

Notably, Equations 6, 7 only consider the real part of the result. Therefore, there is no need to modify the subsequent MLP sub-layer or output layer to handle complex numbers.

3.4 Wavelet transform

Wavelet transform (WT) is developed based on Fourier transform to overcome the limitation of Fourier transform in capturing local features in the spatial domain.

The discrete 1D-WT along the sequence dimension to extract sparse visual features is defined by function $\mathcal{W}_{\text{seq}}$:

$$\boldsymbol{x}'_n = \mathcal{W}_{\text{seq}}(\boldsymbol{x}_n) = \frac{1}{\sqrt{N}}\left[\boldsymbol{T}_\varphi(0,0) * \varphi(\boldsymbol{x}_n) + \sum_{j=0}^{J-1}\sum_{k=0}^{2^j-1} \boldsymbol{T}_\psi(j,k) * \psi_{j,k}(\boldsymbol{x}_n)\right], \tag{8}$$

$$\boldsymbol{T}_\varphi(0,0) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} \boldsymbol{x}_k * \varphi(\boldsymbol{x}_k), \quad \boldsymbol{T}_\psi(j,k) = \frac{1}{\sqrt{N}}\sum_{k'=0}^{N-1} \boldsymbol{x}_{k'} * \psi_{j,k}(\boldsymbol{x}_{k'}), \tag{9}$$

where $n = 0, ..., N-1$, N is typically a power of 2, $*$ denotes element-wise multiplication, $\boldsymbol{T}_\varphi(0,0)$ are the approximation coefficients, $\boldsymbol{T}_\psi(j,k)$ are the detail coefficients, j represents the current scale of wavelet transform with values ranging from 0 to $J-1$, and k denotes the specific position index of the detail transform. $\varphi(x)$ is the scaling function, and $\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)$ is the wavelet function. Here, we use the Haar scaling function and Haar wavelet function for example, which is defined by the equation:

$$\varphi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}, \quad \psi(x) = \begin{cases} 1 & 0 \le x < 0.5 \\ -1 & 0.5 \le x < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

Similarly, the discrete 2D-WT along the feature and sequence dimensions is defined by function $\mathcal{W}_{\text{seq}}(\mathcal{W}_{\text{f}})$:

$$\boldsymbol{x}'_n = \mathcal{W}_{\text{seq}}(\mathcal{W}_{\text{f}}(\boldsymbol{x}_n)), n = 0, ..., N-1, \tag{11}$$

In the subsequent experimental section, we also delve into the exploration of different basis functions as well as their potential combinations.

4 Method

Following a standard vision transformer architecture, the vanilla spikformer incorporates several key components, including the spiking patch splitting (SPS) module, spikformer encoder layers, and a classification head for visual classification tasks. Here, we directly replace vanilla SSA head with the FW head to efficiently manage spike-form features.

In the following sections, we provide an overview of our proposed FWformer in Figure 1, followed by a detailed explanation of the FW head. Finally, we compare the time complexity of both of these two heads.

4.1 Overall architecture

We provide Figure 1 for an overview of our FWformer. First, for a given 2D image sequence $\boldsymbol{I} \in \mathbb{R}^{T\times C\times H\times W}$. In the event-based video datasets, the data shape is $\boldsymbol{I} \in \mathbb{R}^{T\times C\times H\times W}$, where T, C, H, and W denote the time step, channel, height, and width, respectively. In static datasets, a 2D image $\boldsymbol{I}_s \in \mathbb{R}^{C\times H\times W}$ needs to be repeated T times to form an image sequence. The goal of the spiking patch splitting (SPS) module is to linearly project it into a D-dimensional spike-form feature and split this feature into a sequence of N flattened spike-form patches $\boldsymbol{P} \in \mathbb{R}^{T\times N\times D}$. Following the approach of the vanilla spikformer, the SPS module employs convolution operations to introduce inductive bias (Xiao et al., 2021).

Second, to generate spike-form relative position embedding (RPE), the conditional position embedding (CPE) generator (Chu et al., 2021) is utilized in the same manner as the spikformer. The RPE is then added to the patch sequence $\boldsymbol{P}$, resulting in $\boldsymbol{X}_0 \in \mathbb{R}^{T\times N\times D}$.

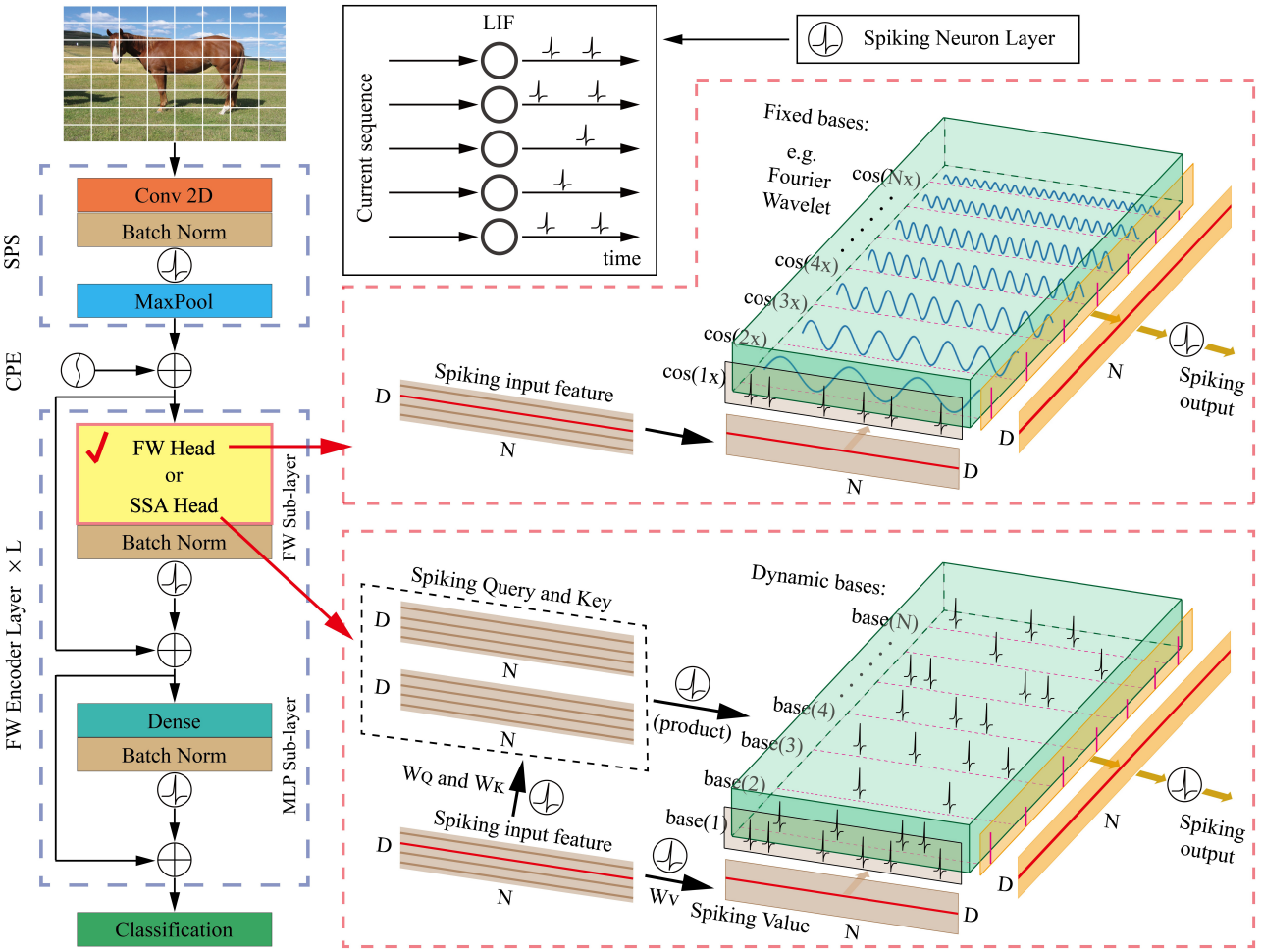

FIGURE 1
The overall architecture of our proposed FWformer. It mainly consists of three components: (1) spiking patch splitting (SPS) module, (2) FWformer encoder layer, and (3) classification layer. Additionally, we highlight the similarities between the FW head and SSA head at a single time step, which inspires us to choose the former as an exploration for more efficient calculations within the spike-form framework.

Third, the L-layer FW encoder is designed to manage $\boldsymbol{X}_0$. Different from spikformer encoder layer with SSA head, our FW encoder layer consists of an FW sub-layer and an MLP sub-layer, both with batch normalization and spiking neuron layer. Residual connections are also applied to both the modules. The FW head in FW sub-layer serves as a critical component in our encoder layer, providing an efficient method for spike-form sparse representation. We have provided two implementations for FW head, including Fourier transform (FT) and wavelet transform (WT). Many works in the past have used FT and WT to alternate between the spatial and frequency domains, allowing for efficient analysis of signals. While in this article we treat them as structured basis functions with prior knowledge for information transformation. These implementations will be thoroughly analyzed in the next section.

Finally, following the processing in spikformer, a global average-pooling (GAP) operation is applied to the resulting spike features, generating a D-dimensional feature. The feature is then fed into the classification module consisting of a spiking fully connected (SFC) layer, which produces the prediction $\boldsymbol{Y}$. The formulation of our FWformer can be expressed as follows:

$$\boldsymbol{P} = \mathrm{SPS}\left(\boldsymbol{I}\right), \tag{12}$$

$$\mathrm{RPE} = \mathrm{CPE}(\boldsymbol{P}), \tag{13}$$

$$\boldsymbol{X}_0 = \boldsymbol{P} + \mathrm{RPE}, \tag{14}$$

$$\boldsymbol{X}'_l = \mathcal{SN}(\mathrm{BN}(\mathrm{FW}(\boldsymbol{X}_{l-1}))) + \boldsymbol{X}_{l-1}, \tag{15}$$

$$\boldsymbol{X}_l = \mathcal{SN}(\mathrm{BN}(\mathrm{MLP}(\boldsymbol{X}'_l))) + \boldsymbol{X}'_l, \tag{16}$$

$$\boldsymbol{Y} = \mathrm{SFC}(\mathrm{GAP}(\boldsymbol{X}_L)), \tag{17}$$

where $\boldsymbol{I} \in \mathbb{R}^{T\times C\times H\times W}$, $\boldsymbol{P} \in \mathbb{R}^{T\times N\times D}$, $\mathrm{RPE} \in \mathbb{R}^{T\times N\times D}$, $\boldsymbol{X}_0 \in \mathbb{R}^{T\times N\times D}$, $\boldsymbol{X}'_l \in \mathbb{R}^{T\times N\times D}$, $\boldsymbol{X}_l \in \mathbb{R}^{T\times N\times D}$ and $l = 1, ..., L$.

Moreover, the membrane shortcut (MS), which has been applied in many existing works (Chen et al., 2023; Yao et al., 2024), is also utilized in our model for comparison. It establishes a shortcut between the membrane potential of spiking neurons in various layers to enhance performance and increase biological plausibility (Yao et al., 2024).

4.2 The FW head

Given input spike sequences $\boldsymbol{X} \in \mathbb{R}^{T\times N\times D}$, these features are then transformed into spiking sequences $\boldsymbol{X}' \in \mathbb{R}^{T\times N\times D}$ through a $\mathcal{SN}$ layer. The formulation can be expressed as:

$$\begin{aligned} \mathrm{FW}(\boldsymbol{X}) &= \mathrm{FT}(\boldsymbol{X})\ or\ \mathrm{WT}(\boldsymbol{X}),\\ \boldsymbol{X}' &= \mathcal{SN}(\mathrm{BN}((\mathrm{FW}(\boldsymbol{X})))), \end{aligned} \tag{18}$$

In contrast to the SSA head in Equation 5, the FW head does not involve any learnable parameters or Self-Attention calculations. Here, we can choose from Fourier transform (FT) and wavelet transform (WT) with fixed basis functions. We can also combine different wavelet bases to form a superior function, which is defined as follows:

$$\begin{aligned} Base &= a\cdot Base1 + b\cdot Base2 + c\cdot Base3,\\ \mathrm{FW}(\boldsymbol{X}) &= Base(\boldsymbol{X}), \end{aligned} \tag{19}$$

where a, b, and c are learnable parameters, and $Base1$, $Base2$, and $Base3$ are selected bases. Since wavelet transform is a linear transformation, Equation 19 can also be written as:

$$\mathrm{FW}(\boldsymbol{X}) = a\cdot Base1(\boldsymbol{X}) + b\cdot Base2(\boldsymbol{X}) + c\cdot Base3(\boldsymbol{X}) \tag{20}$$

Further analysis and experiments will be conducted based on the proposed FW head in the following sections.

4.3 Time complexity analysis of FW vs. SSA

We make a time complexity analysis between SSA, Fourier transform (FT), and wavelet transform (WT). The results are presented in Table 1. In the subsequent experimental section, we also conduct a more specific comparison of the training and inference speeds between FW and SSA under the same conditions.

In SSA (Equation 5), since there is no softmax operation, the order of calculation between $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ can be changed: either $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$ followed by $\boldsymbol{V}$, or $\boldsymbol{K}^{\mathrm{T}}\boldsymbol{V}$ followed by $\boldsymbol{Q}$. The former has a time complexity of $\mathcal{O}(N^2d)$, while the latter has $\mathcal{O}(Nd^2)$, where d is the feature dimension per head.In practice, the SSA in Equation 5 can be extended to multi-head SSA. In this case, $d = D/H$, where H is the number of heads. The second complexity, $\mathcal{O}(Nd^2)$, cannot be simply considered as $\mathcal{O}(N)$ due to the large constant d^2 involved. Only when the sequence length N is significantly larger than the feature dimension per head d does it demonstrate a significant computational efficiency advantage over the first complexity, $\mathcal{O}(N^2d)$.

In our implementation, we utilize the fast Fourier transform (FFT) algorithm to compute the discrete FT. Specifically, we employ the Cooley–Tukey algorithm (Cooley and Tukey, 1965), which recursively expresses the discrete FT of a sequence of length $N = N_1N_2$ in terms of N_1 smaller discrete FTs of size N_2, reducing the time complexity to $\mathcal{O}(N\log N)$ for discrete 1D-FT along the sequence dimension. Similarly, for discrete 2D-FT first along the feature dimension and then along the sequence dimension, the time complexity is $\mathcal{O}(D\log D) + \mathcal{O}(N\log N)$. In general, the complexity of WT is comparable to that of FFT (Gonzales and Wintz, 1987).

5 Experiments

We conduct experiments on event-based video datasets (CIFAR10-DVS and DvsGesture), as well as static image datasets (CIFAR10 and CIFAR100). The FWformer is trained from scratch and compared with existing methods, including spikformer with SSA and its variant. More analyses are also given about the effects of different wavelet bases and their combinations.

5.1 Experiment settings

To ensure a fair comparison, we ensure the same configurations of spikformer with SSA for datasets, implementation details, and evaluation metrics. To conduct the experiments, we implement the models using PyTorch and SpikingJelly (Fang et al., 2023). All experiments are conducted on NVIDIA A100 GPU.

5.1.1 Event-based video datasets

For the CIFAR10-DVS and DvsGesture datasets, which have an image size of 128×128, we employ the spiking patch splitting (SPS) module with a patch size of 16×16. This configuration splits each image into a sequence with a length N of 64 and a feature dimension D of 256. We utilize 2 FWformer encoder layers and set the time step of the spiking neuron to 16. The training process consists of 106 epochs for CIFAR10-DVS and 200 epochs for DvsGesture. We employ the AdamW optimizer with a batch size of 16. The learning rate is initialized to 0.1 and reduced using cosine decay. Additionally, data augmentation techniques, as described in Li et al. (2022), are applied specifically to the CIFAR10-DVS dataset.

5.1.2 Static image datasets

For the CIFAR10/100 datasets featuring an image size of 32×32, we employ the SPS module with a patch size of 4×4, which splits each image into a sequence of length $N = 64$ and a feature dimension of $D = 384$. For the FWformer Encoder, we use four layers, and the time-step of the spiking neuron is set to 4. During training, we utilize the AdamW optimizer with a batch size of 128. The training process spans 400 epochs, with a cosine-decay learning rate starting at 0.0005. Following the approach outlined in Yuan

TABLE 1 The time complexity for different methods.

Methods	SSA (Zhou et al., 2023)	1D-FFT	2D-FFT	2D-WT
Time complexity	$\mathcal{O}(N^2d)$ or $\mathcal{O}(Nd^2)$	$\mathcal{O}(N\log N)$	$\mathcal{O}(D\log D) + \mathcal{O}(N\log N)$	$\mathcal{O}(D\log D) + \mathcal{O}(N\log N)$

We have $N = 64$, $D = 384$ or 256, and $d = 32$.

TABLE 2 Accuracy performance comparison of our method with existing methods on CIFAR10-DVS (DVS10), DvsGesture (DVS128), CIFAR10, and CIFAR100.

Methods	Architecture	Time step (DVS10/128)	Top-1 acc. (DVS10/128)
LIAF (Wu et al., 2021)	LIAF-Net	10/60	70.4/97.6
TA-SNN (Yao et al., 2021)	TA-SNN	10/60	72.0/98.6
Rollout (Kugele et al., 2020)	–	48/240	66.8/97.2
DECOLLE (Kaiser et al., 2020)	–	– /500	– /95.5
tdBN (Zheng et al., 2021)	ResNet-19	10/40	67.8/96.9
PLIF (Fang et al., 2021a)	–	20/20	74.8/97.6
D-ResNet (Fang et al., 2021b)	Wide-7B-Net	16/16	74.4/97.9
Dspike (Li et al., 2021)	–	10/ –	75.4/–
SALT (Kim and Panda, 2021)	–	20/ –	67.1/–
DSR (Meng et al., 2022)	–	10/ –	77.3/–
SDSA (Yao et al., 2024)	Spikformer-2-256	16/16	80.0/99.3
SSA (Zhou et al., 2023)	Spikformer-2-256	16/16	80.9/98.3
1D-FFT	FWformer-2-256	16/16	80.5/99.0
2D-FFT	FWformer-2-256	16/16	80.6/98.4
2D-WT-Haar	FWformer-2-256	16/16	81.0/98.5
1D-FFT*	FWformer-2-256	16/16	80.8/**99.5**
2D-FFT*	FWformer-2-256	16/16	80.7/98.2
2D-WT-Haar*	FWformer-2-256	16/16	**81.2**/99.1
Methods	**Architecture**	**Time step**	**Top-1 acc. (CIFAR10/100)**
Hybrid training (Rathi et al., 2020)	VGG-11	125	92.22/67.87
Diet-SNN (Rathi and Roy, 2020)	ResNet-20	10/5	92.54/64.07
STBP (Wu et al., 2018)	CIFARNet	12	89.83/–
STBP NeuNorm (Wu et al., 2019)	CIFARNet	12	90.53/–
Dspike (Li et al., 2021)	–	6	94.3/74.2
TSSL-BP (Zhang and Li, 2020)	CIFARNet	5	91.41/–
STBP-tdBN (Zheng et al., 2021)	ResNet-19	4	92.92/70.86
TET (Deng et al., 2022)	ResNet-19	4	94.44/74.47
ANN methods	ResNet-19	1	94.97/75.35
	transformer-4-384	1	**96.73/81.02**
SDSA (Yao et al., 2024)*	Spikformer-4-384	4	**95.6/78.4**
SSA (Zhou et al., 2023)	Spikformer-4-384	4	95.51/78.21
1D-FFT	FWformer-4-384	4	94.9/77.3
2D-FFT	FWformer-4-384	4	95.1/77.9
2D-WT-Haar	FWformer-4-384	4	95.2/78.1
1D-FFT*	FWformer-4-384	4	95.5/78.0
2D-FFT*	FWformer-4-384	4	95.0/78.3
2D-WT-Haar*	FWformer-4-384	4	**95.6**/78.2

Our FWformer [∗ means replacing vanilla residual connection with Membrane Shortcut (MS)] outperforms spikformer with SSA on event-based video datasets in terms of Top-1 acc. and achieves comparable accuracy on static datasets (the text in bold indicates the best results). It is necessary to mention that in the Architecture column, the FWformer-L-D refers to a configuration with L layers of FW encoder layer and a feature dimension of D, while the spikformer-L-D represents L layers of spikformer encoder block with a feature dimension of D. For event-based video datasets (DVS10, DVS128), L is set to 2 and D to 256, whereas for static image datasets (CIFAR10, CIFAR100), L is set to 4 and D to 384. For each dataset, all hyperparameters including learning rate, training epochs, time steps, and optimizer settings, are kept identical for both spikformer and FWformer, ensuring a fair comparison, as detailed in Section 5.1.

TABLE 3 Memory usage and speed performance comparison of our method with existing methods on CIFAR10-DVS (DVS10), DvsGesture (DVS128) and CIFAR-static (CIFAR10 and CIFAR100).

Methods	Param (M)	Memory (DVS10/128) (GB)	Training speed (DVS10/128) (ms/batch)	Inference speed (DVS10/128) (ms/batch)
STBP-tdBN (Zheng et al., 2021)	12.63	25.86/25.87	65/194	27/98
TET (Deng et al., 2022)	12.63	36.13/36.17	71/203	22/77
SDSA (Yao et al., 2024)	2.59	9.02/9.03	73/245	29/101
SSA (Zhou et al., 2023)	2.59	9.02/9.03	76/246	30/105
1D-FFT	**2.06**	8.67/**8.71**	**51/121**	**11/32**
2D-FFT	**2.06**	**8.54**/8.74	55/135	21/37
2D-WT-Haar	**2.06**	8.70/8.73	62/139	21/46
DWT-C	**2.06**	8.55/8.74	69/158	23/48
Methods	**Param (M)**	**Memory (CIFAR-static) (GB)**	**Training Speed (CIFAR-static) (ms/batch)**	**Inference Speed (CIFAR-static) (ms/batch)**
STBP-tdBN (Zheng et al., 2021)	12.63	8.02	155	20
TET (Deng et al., 2022)	12.63	8.19	148	23
SDSA (Yao et al., 2024)	9.32	11.69	162	33
SSA (Zhou et al., 2023)	9.32	11.69	166	31
1D-FFT	**6.96**	**8.61**	**118**	**12**
2D-FFT	**6.96**	8.75	122	13
2D-WT-Haar	**6.96**	9.33	121	19
DWT-C	**6.96**	9.86	136	25

Our FWformer outperforms spikformer with SSA (Zhou et al., 2023) and its variant (Yao et al., 2024) when comparing GPU memory usage, training speed and inference speed under identical operating conditions. It is important to mention that the model architectures and hyperparameters are consistent with those of the models listed in Table 2.

et al. (2021), we apply standard data augmentation techniques such as random augmentation, mixup, and cutmix during training.

5.2 Accuracy performance

We evaluate the accuracy performance on visual classification tasks, utilizing Top-1 accuracy (Top-1 acc.) as the performance metric. The results of our FWformer, spikformer with SSA, and other existing methods (both SNNs and ANNs, including the spikformer variant) (Yao et al., 2024) on event-based video datasets as well as static image datasets are presented in Table 2.

Our FWformer achieves remarkable accuracy, reaching 81.2% on CIFAR10-DVS with 2D-WT-Haar, and an impressive 99.5% on DvsGesture with 1D-FFT. The performances surpass the spikformer with SSA by 0.3 and 1.2%, respectively. While on static datasets, our FWformer variants demonstrate comparable Top-1 accuracy. The results demonstrate the advantage of our methods, particularly on event-based video datasets.

5.3 Computational costs and speed performance

Furthermore, we conduct a comprehensive comparison between existing works and our FWformer in terms of GPU memory usage, training speed, and inference speed, ensuring identical operating conditions. The training speed represents the time taken for the forward and back-propagation of a batch of data, while the inference speed denotes the time taken for the forward-propagation of a batch of data in milliseconds (ms). To minimize variance, we calculate the average time spent over 100 batches. The results are presented in Table 3 (DWT-C means 2D-WT combination with learnable parameters, which will be discussed in the next subsection).

In the case of event-based video datasets, our FWformer achieves a significant reduction in the number of parameters, ∼20%, under identical hyperparameter configurations and operating conditions. This reduction, attributed to the absence of learnable parameters, translates to around 4%–5% memory savings. Moreover, our FWformer demonstrates remarkable improvements in both training and inference speeds, showing increases of ∼ 9%–51% and 33%–70%, respectively, compared to SSA. While in the case of static datasets, our FWformer also shows several advantages under identical hyperparameter configurations and operating conditions. It achieves a notable reduction in the number of parameters, ∼25%, leading to memory savings of nearly 26%. Furthermore, our FWformer enhances both training and inference speeds by ∼18%–29% and 19%–61%, respectively.

We also provide an analysis of energy efficiency. We estimate the theoretical energy consumption of FWformer mainly according to Yao et al. (2024), Horowitz (2014), and Zhou et al. (2023). It is

TABLE 4 The theoretical energy consumption on DVS10 (CIFAR10-DVS), DVS128 (DvsGesture), CIFAR10, and CIFAR100 dataset.

Methods	DVS10 OPs (G)/Power (mJ)	DVS128 OPs (G)/Power (mJ)
SDSA (Yao et al., 2024)	1.561/0.816	1.620/0.713
SSA (Zhou et al., 2023)	1.852/0.943	1.914/0.822
1D-FFT	1.547/0.752	1.608/0.650
2D-FFT	1.548/0.752	1.609/0.653
2D-WT	1.549/0.753	1.609/0.651
DWT-C	1.553/0.753	1.613/0.652
Methods	**CIFAR10 OPs (G)/Power (mJ)**	**CIFAR100 OPs (G)/Power (mJ)**
SDSA (Yao et al., 2024)	0.951/0.415	1.446/0.609
SSA (Zhou et al., 2023)	1.186/0.523	1.737/0.748
1D-FFT	0.942/0.392	1.438/0.578
2D-FFT	0.943/0.393	1.438/0.584
2D-WT	0.944/0.393	1.439/0.584
DWT-C	0.947/0.394	1.442/0.586

DWT-C means 2D-WT combination with learnable parameters.

TABLE 5 Accuracy performance of different wavelet bases as well as their combination with learnable (DWT-C) or fixed (DWT-C-F) parameters on DVS10 and DVS128 datasets [∗ means replacing vanilla residual connection with Membrane Shortcut (MS)].

Methods	Architecture	Time step DVS10/128	Top-1 acc. DVS10/128
2D-WT-Haar	FWformer-2-256	16/16	81.0/98.5
2D-WT-Haar*	FWformer-2-256	16/16	81.2/99.1
Db1	FWformer-2-256	16/16	81.0/98.7
Bior1.1	FWformer-2-256	16/16	80.9/98.2
Rbio1.1	FWformer-2-256	16/16	80.4/98.1
DWT-C-F	FWformer-2-256	16/16	80.6/98.2
DWT-C-F*	FWformer-2-256	16/16	81.1/98.9
DWT-C	FWformer-2-256	16/16	**81.3**/99.1
DWT-C*	FWformer-2-256	16/16	81.2/**99.8**

The text in bold indicates the best results.

calculated by the following two equations:

$$\mathrm{SOPs}(l) = Rate \times T \times \mathrm{FLOPs}(l), \tag{21}$$

$$E_{FWformer} = E_{MAC} \times \mathrm{EL}^1_{Conv} + E_{AC} \times \left(\sum_{k=2}^{K} \mathrm{SOP}^k_{Conv} + \sum_{m=1}^{M} \mathrm{SOP}^m_{FC} + \sum_{n=1}^{N} \mathrm{SOP}^n_{FW}\right), \tag{22}$$

SOPs(l) means synaptic operations [the number of spike-based accumulate (AC) operations] of layer l, *Rate* is the average firing rate of input spike train to layer l, T is the time window of LIF neurons, and FLOPs(l) refers to the floating point operations [the number of multiply-and-accumulate (MAC) operations] of layer l. We assume that the MAC and AC operations are implemented on the 45nm hardware (Horowitz, 2014), with $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$.

EL^1_{Conv} represents the FLOPs of convolution module in ANNs. It is used for the first layer to convert static images into spike trains, which can also be written as SOP^1_{Conv} for event-based video datasets, and SOP^l_{Net} (SOP^k_{Conv}, SOP^m_{FC}, SOP^n_{FW}) is for the rest of FWformer.

The experimental settings are the same as in the main text. Each cell in Table 4 contains results presented in the form of OPs(G)/Power(mJ), where OPs refers to the total SOPs in a SNN model, and Power refers to the average theoretical energy consumption when predicting one sample from the datasets.

The results indicate that our methods can achieve a reduction in energy consumption of ∼20%–25% compared to SSA (Zhou et al., 2023) and 4%–9% compared to its variant (Yao et al., 2024). This is primarily due to lower computational complexity of the FW head, as reflected in fewer total SOPs (OPs). Our FWformer demonstrates enhanced energy efficiency.

5.4 From orthogonal to non-orthogonal bases

In the previous experiments, the Haar base was used as the default choice for wavelet transform. We have also compared the performance of some other wavelet bases including Db1, Bior1.1, and Rbio1.1, each having different functions for deconstructing the spike-form feature while maintaining orthogonality. The results on CIFAR10-DVS and DvsGesture are presented in Table 5. Interestingly, most alternative basis functions yield similar Top-1 accuracy. Their performance is comparable to or even better than that of spikformer. It is essential to highlight that wavelet transform offers numerous different basis function options, and our exploration has not been exhaustive. Investigating the influence of more basis function choices on accuracy, as well as the possibility of identifying superior basis functions, is an avenue for future research.

However, a more interesting question arises: Is it always necessary to pursue orthogonality? Although in many cases, orthogonality signifies sparse and efficient information representation, neural networks may show the opposite phenomenon in the actual training process, that is, parameters naturally tend toward overlapping representations, and SSA is no exception. To illustrate this, we treat SSA as basis functions as proposed in the previous sections, and then quantitatively measure changes in their orthogonality during training. We calculate the inner product of each row vector (one base) in $Q \times K$ with others (other bases) and sum them in each training step. The variation trend is shown in Figure 2A. Initially, network parameters are nearly orthogonal at initialization, but their orthogonality is continuously decreasing during training. A diagram visualizing how the basis functions change during training is provided in Figure 2B. Inspired by this phenomenon, we further explore the combination of different wavelet bases to form fixed non-orthogonal basis functions, as depicted in Figure 2C. We assume

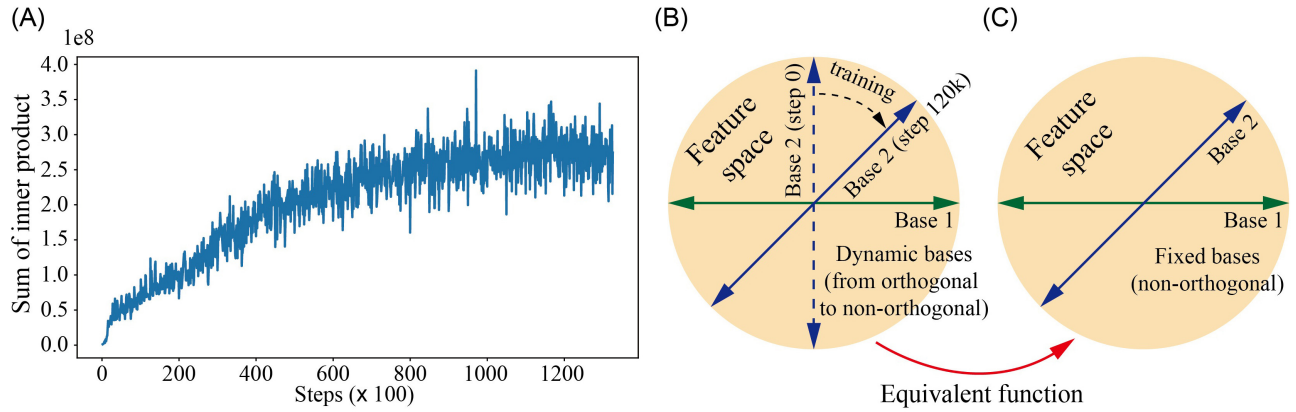

FIGURE 2
(A) We treat spiking self-attention as a set of basis functions and proceed to measure the changes in their orthogonality throughout the training process. **(B)** A diagram visualizing how the basis functions, spanning a feature space, are transformed from orthogonal to non-orthogonal, with only two axes used for simplification. **(C)** A diagram visualizing our endeavor to employ fixed non-orthogonal bases.

TABLE 6 Ablation studies examining the impact of varying layer configurations on accuracy.

Methods	Layer number	Top-1 acc. (DVS128)
No FW or SSA	–	95.4
Spikformer SSA (Zhou et al., 2023)	1	97.2
	2	98.3
	3	98.3
	4	97.6
FWformer 2D-WT-Haar*	1	98.2
	2	99.1
	3	99.1
	4	98.2
FWformer 1D-FFT*	1	99.7
	2	99.5
	3	98.4
	4	98.4
FWformer DWT-C*	1	98.6
	2	99.8
	3	99.9
	4	99.0

∗ Means we replace vanilla residual connection with Membrane Shortcut (MS).

that the pre-trained dynamic bases may serve an equivalent function to the fixed non-orthogonal bases.

Here we choose Bior1.1, Haar, and Db1 for further exploration. In practice, we introduced dynamics to the coefficients of different wavelet bases using the following formulation: $a = p_1^1$, $b = p_2^2$, $c = p_3^3$. To search for the optimal coefficients for their combinations, we initialized the learnable parameters p_1, p_2 and p_3 at 0.5 and then conducted training. The new FW head will have three parameters to learn, which exert minimal influence on the overall computation of the network but play a crucial role in finding suitable combinations. After training, these parameters were optimized to 0.9683, 1.0895, and 1.2445, respectively, demonstrating a joint optimization process with the network parameters. The results are presented in Table 5. Consistent with our hypothesis, the use of fixed non-orthogonal basis functions further improves accuracy performance (0.4%–1.5% improvements on event-based video datasets, compared to vanilla spikformer). We have also set fixed values of p_1, p_2, and p_3 to 0.5 (DWT-C-F), the results were inferior to DWT-C and resembled the performance of using only a single wavelet base. This highlights the effectiveness of introducing learnable parameters. A possible explanation is that manually tuning these parameters would involve an excessively large search space, whereas the learnable approach significantly improves search efficiency.

We attempt to conduct a preliminary analysis of the situations in which our FWformer is applicable: In contrast to conventional signal processing, complex tasks such as Natural Language Processing (NLP) and Automatic Speech Recognition (ASR) need the designed models to learn diverse syntactic and semantic relationships, which can hardly be represented simply by fixed basis functions such as Fourier bases. For this reason, the network has to form dynamic higher-order basis functions, which are adjusted by not only the changing inputs but also the parameter learning of the network itself. We can regard these basis functions in networks as hyper-parameters that need continuous adjustment. However, this also means each time the basis functions change, the rest of the network has to adapt accordingly, which is understandable in complex tasks but not necessary in some other cases (e.g. event-based video tasks). Moreover, within spike-form frameworks, the features are represented by such sparse spiking signals that the correlation between them is too weak to form closed similarity, so it is more suitable to use structured fixed basis functions (e.g. Fourier bases and wavelet bases) containing prior knowledge to get a simplified network.

5.5 Ablation studies

In this section, we conduct ablation studies on the impact of different layers of FW encoder layer or SSA block. The experiments were conducted on the DVS128 dataset, and the results are presented in Table 6. The hyperparameters are consistent with those of the models listed in Table 2.

The results reveal that while adding layers of FW encoder layer can enhance accuracy, there is an optimal point beyond which increasing the number of layers does not yield better results. This trend also holds true for SSA layers. Within the spike-form frameworks, too many layers might lead to overfitting. Overall, an optimal performance is achieved with two FW layers for the DVS128 dataset.

6 Conclusion

We present the FWformer that replaces SSA with spike-form FW head, based on the hypothesis that both of them use dynamic or fixed bases to transform information. The proposed model achieves comparable or better accuracy, higher training and inference speed, and reduced computational cost, on both event-based video datasets and static datasets. We analyze the orthogonality in SSA during training and assume that the pre-trained dynamic bases serve an equivalent function to the fixed bases, which inspires us to explore non-orthogonal combined bases and get even higher accuracy. Additionally, we provide an analysis of why and under what scenarios our FWformer is effective, indicating the promising refinement of new transformers in the future, which is inspired by biological discovery and information theory.

Data availability statement

Publicly available datasets were analyzed in this study. The CIFAR10-DVS dataset can be downloaded from: https://figshare.com/articles/dataset/CIFAR10-DVS_New/4724671. The DvsGesture dataset can be downloaded from: http://research.ibm.com/dvsgesture/. The CIFAR10 and CIFAR100 dataset can be downloaded from: https://www.cs.toronto.edu/~kriz/cifar.html.

Author contributions

QW: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. DZ: Formal analysis, Resources, Supervision, Writing – original draft. XC: Funding acquisition, Resources, Supervision, Writing – review & editing. TZ: Funding acquisition, Resources, Supervision, Validation, Writing – review & editing. BX: Funding acquisition, Resources, Supervision, Validation, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was supported by the Research on Brain-Inspired Auditory Front-End Models and Systems (Grant no. 2021ZD0201500).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

References

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., et al. (2020). "End-to-end object detection with transformers," in *Computer Vision-ECCV 2020: 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I 16* (Cham: Springer), 213–229. doi: 10.1007/978-3-030-58452-8_13

Chen, G., Peng, P., Li, G., and Tian, Y. (2023). Training full spike neural networks via auxiliary accumulation pathway. *arXiv* [Preprint]. arXiv:2301.11929. doi: 10.48550/arXiv.2301.11929

Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., et al. (2021). "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Nashville, TN: IEEE), 12299–12310. doi: 10.1109/CVPR46437.2021.01212

Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., et al. (2021). Twins: revisiting the design of spatial attention in vision transformers. *Adv. Neural Inf. Process. Syst.* 34, 9355–9366.

Cooley, J. W., and Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* 19, 297–301. doi: 10.1090/S0025-5718-1965-0178586-1

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Dayan, P., and Abbott, L. F. (2005). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA: MIT press.

Deng, S., Li, Y., Zhang, S., and Gu, S. (2022). Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv* [Preprint]. arXiv:2202.11946. doi: 10.48550/arXiv.2202.11946

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2020). "An image is worth 16x16 words: transformers for image recognition at scale," in *International Conference on Learning Representations*.

Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., et al. (2023). Spikingjelly: an open-source machine learning infrastructure platform for spike-based intelligence. *Sci. Adv*. 9:eadi1480. doi: 10.1126/sciadv.adi1480

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., Tian, Y., et al. (2021b). Deep residual learning in spiking neural networks. *Adv. Neural Inf. Process. Syst*. 34, 21056–21069.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., Tian, Y., et al. (2021a). "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (Montreal, QC: IEEE), 2661–2671. doi: 10.1109/ICCV48922.2021.00266

Gonzales, R. C., and Wintz, P. (1987). *Digital Image Processing*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Han, B., Srinivasan, G., and Roy, K. (2020). "RMP-SNN: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Seattle, WA: IEEE), 13558–13567. doi: 10.1109/CVPR42600.2020.01357

Horowitz, M. (2014). "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)* (San Francisco, CA: IEEE), 10–14. doi: 10.1109/ISSCC.2014.6757323

Hu, Y., Tang, H., and Pan, G. (2021). "Spiking deep residual networks," in *IEEE Transactions on Neural Networks and Learning Systems*.

Izhikevich, E. M., Gally, J. A., and Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cereb Cortex* 14, 933–944. doi: 10.1093/cercor/bhh053

Kaiser, J., Mostafa, H., and Neftci, E. (2020). Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci*. 14:424. doi: 10.3389/fnins.2020.00424

Katharopoulos, A., Vyas, A., and Pappas, N. F. (2020). "Transformers are RNNs: fast autoregressive transformers with linear attention," in *International Conference on Machine Learning* (PMLR), 5156–5165.

Kim, Y., and Panda, P. (2021). Optimizing deeper spiking neural networks for dynamic vision sensing. *Neural Netw*. 144, 686–698. doi: 10.1016/j.neunet.2021.09.022

Kugele, A., Pfeil, T., Pfeiffer, M., and Chicca, E. (2020). Efficient processing of spatio-temporal data streams with spiking neural networks. *Front. Neurosci*. 14:439. doi: 10.3389/fnins.2020.00439

Lee, C., Sarwar, S. S., Panda, P., Srinivasan, G., and Roy, K. (2020). Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci*. 14:119. doi: 10.3389/fnins.2020.00119

Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontanon, S. (2021). FNET: mixing tokens with Fourier transforms. *arXiv* [Preprint]. arXiv:2105.03824. doi: 10.48550/arXiv.2105.03824

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., Gu, S., et al. (2021). Differentiable spike: rethinking gradient-descent for training spiking neural networks. *Adv. Neural Inf. Process. Syst*. 34, 23426–23439.

Li, Y., Kim, Y., Park, H., Geller, T., and Panda, P. (2022). "Neuromorphic data augmentation for training spiking neural networks," in *Computer Vision-ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part VII* (Cham: Springer), 631–649. doi: 10.1007/978-3-031-20071-7_37

Lotfi Rezaabad, A., and Vishwanath, S. (2020). "Long short-term memory spiking networks and their applications," in *International Conference on Neuromorphic Systems 2020* (New York, NY: ACM), 1–9. doi: 10.1145/3407197.3407211

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Netw*. 10, 1659–1671. doi: 10.1016/S0893-6080(97)00011-7

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., Lu, Z.-Q., et al. (2022). "Training high-performance low-latency spiking neural networks by differentiation on spike representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12444–12453. doi: 10.1109/CVPR52688.2022.01212

Mueller, E., Studenyak, V., Auge, D., and Knoll, A. (2021). "Spiking transformer networks: a rate coded approach for processing sequential data," in *2021 7th International Conference on Systems and Informatics (ICSAI)* (Chongqing: IEEE), 1–5. doi: 10.1109/ICSAI53574.2021.9664146

Rathi, N., and Roy, K. (2020). Diet-SNN: direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv* [Preprint]. arXiv:2008.03658. doi: 10.48550/arXiv.2008.03658

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. (2020). Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv* [Preprint]. arXiv:2005.01807. doi: 10.48550/arXiv.2005.01807

Rueckauer, B., Lungu, I. A., Hu, Y., Pfeiffer, M., and Liu, S. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci*. 11:682. doi: 10.3389/fnins.2017.00682

Shrestha, S. B., and Orchard, G. (2018). Slayer: Spike layer error reassignment in time. *Adv. Neural Inf. Process. Syst*. 31.

Tang, G., Kumar, N., Yoo, R., and Michmizos, K. (2021). "Deep reinforcement learning with population-coded spiking neural network for continuous control," in *Conference on Robot Learning* (PMLR), 2016–2029.

Tian, C., Zheng, M., Zuo, W., Zhang, B., Zhang, Y., Zhang, D., et al. (2023). Multi-stage image denoising with the wavelet transform. *Pattern Recognit*. 134:109050. doi: 10.1016/j.patcog.2022.109050

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Adv. Neural Inf. Process. Syst*. 30.

Wang, Q., Zhang, T., Han, M., Wang, Y., Zhang, D., Xu, B., et al. (2023). Complex dynamic neurons improved spiking transformer network for efficient automatic speech recognition. *Proc. AAAI Conf. Artif. Intell*. 37, 102–109. doi: 10.1609/aaai.v37i1.25081

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., et al. (2021). "Pyramid vision transformer: a versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision* (Montreal, QC: IEEE), 568–578. doi: 10.1109/ICCV48922.2021.00061

Wu, Y., Deng, L., Li, G., and Zhu, J. and Shi L (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci*. 12, 331. doi: 10.3389/fnins.2018.00331

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L., et al. (2019). Direct training for spiking neural networks: Faster, larger, better. *Proc. AAAI Conf. Artif. Intell*. 33, 1311–1318. doi: 10.1609/aaai.v33i01.33011311

Wu, Z., Zhang, H., Lin, Y., Li, G., Wang, M., Tang, Y., et al. (2021). Liaf-net: leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Trans. Neural Netw. Learn. Syst*. 33, 6249–6262. doi: 10.1109/TNNLS.2021.3073016

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollar, P., Girshick, R., et al. (2021). Early convolutions help transformers see better. *Adv. Neural Inf. Process. Syst*. 34, 30392–30400.

Xu, M., Wu, Y., Deng, L., Liu, F., Li, G., Pei, J., et al. (2021). "Exploiting spiking dynamics with spatial-temporal feature normalization in graph learning," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, ed. Z. Zhou, 3207–3213. doi: 10.24963/ijcai.2021/441

Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., et al. (2021). "Temporal-wise attention spiking neural networks for event streams classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (Montreal, QC: IEEE), 10221–10230. doi: 10.1109/ICCV48922.2021.01006

Yao, M., Hu, J., Zhou, Z., Yuan, L., Tian, Y., Xu, B., et al. (2024). Spike-driven transformer. *Adv. Neural Inf. Process. Syst*. 36.

You, N., Han, L., Zhu, D., and Song, W. (2023). Research on image denoising in edge detection based on wavelet transform. *Appl. Sci*. 13:1837. doi: 10.3390/app13031837

You, W., Sun, S., and Iyyer, M. (2020). Hard-coded Gaussian attention for neural machine translation. *arXiv* [Preprint]. arXiv:2005.00742. doi: 10.48550/arXiv.2005.00742

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., et al. (2021). "Tokens-to-token VIT: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF international conference on computer vision*, 558–567. doi: 10.1109/ICCV48922.2021.00060

Zhang, M., Tong, X.-J., Liu, J., Wang, Z., Liu, J., Liu, B., et al. (2020). Image compression and encryption scheme based on compressive sensing and Fourier transform. *IEEE Access* 8, 40838–40849. doi: 10.1109/ACCESS.2020.2976798

Zhang, T., Cheng, X., Jia, S., Poo, M. M., Zeng, Y., Xu, B., et al. (2021). Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks. *Sci. Adv*. 7:eabh0146. doi: 10.1126/sciadv.abh0146

Zhang, W., and Li, P. (2020). Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Adv. Neural Inf. Process. Syst*. 33, 12022–12033.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. (2021). Going deeper with directly-trained larger spiking neural networks. *Proc. AAAI Conf. Artif. Intell*. 35, 11062–11070. doi: 10.1609/aaai.v35i12.17320

Zhou, Z., Zhu, Y., He, C., Wang, Y., Yan, S., Tian, Y., et al. (2023). "Spikformer: when Spiking Neural Network Meets Transformer," in *International Conference on Learning Representations*.