This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Decision SpikeFormer: Spike-Driven Transformer for Decision Making

Wei Huang[1,2*]
[1]Shanghai AI Laboratory
huangwe@whu.edu.cn

Qinying Gu[1†]
[2]Wuhan University
guqinying@pjlab.org.cn

Nanyang Ye[3†]
[3]Shanghai Jiao Tong University
ynylincoln@sjtu.edu.cn

Abstract

Offline reinforcement learning (RL) enables policy training solely on pre-collected data, avoiding direct environment interaction—a crucial benefit for energy-constrained embodied AI applications. Although Artificial Neural Networks (ANN)-based methods perform well in offline RL, their high computational and energy demands motivate exploration of more efficient alternatives. Spiking Neural Networks (SNNs) show promise for such tasks, given their low power consumption. In this work, we introduce DSFormer, the first spike-driven transformer model designed to tackle offline RL via sequence modeling. Unlike existing SNN transformers focused on spatial dimensions for vision tasks, we develop Temporal Spiking Self-Attention (TSSA) and Positional Spiking Self-Attention (PSSA) in DSFormer to capture the temporal and positional dependencies essential for sequence modeling in RL. Additionally, we propose Progressive Threshold-dependent Batch Normalization (PTBN), which combines the benefits of LayerNorm and BatchNorm to preserve temporal dependencies while maintaining the spiking nature of SNNs. Comprehensive results in the D4RL benchmark show DSFormer's superiority over both SNN and ANN counterparts, achieving 78.4% energy savings, highlighting DSFormer's advantages not only in energy efficiency but also in competitive performance. Code and models are public at project page.

1. Introduction

Offline reinforcement learning (RL) aims to develop effective policies solely from pre-collected data that capture agent behaviors without interacting with the environment [21]. This approach is crucial for embodied AI applications, particularly when direct exploration is constrained by safety, energy and resource limitations. While artificial neural networks (ANNs) have driven significant advancements in offline RL—addressing challenges from policy regularization [9, 19], value function approximation [18, 20] to conditional sequence modeling (CSM) [3, 16]—they often entail substantial computational and energy costs.

Spiking Neural Networks (SNNs), as the third generation of neural networks [27], offer a promising alternative, bringing low-power, event-driven efficiency to offline RL, especially in energy-constrained, embodied learning scenarios. SNNs have drawn considerable interest for their low energy demands, alignment with biological processes, and compatibility with neuromorphic hardware [4, 22, 30]. Unlike ANNs, SNNs operate efficiently by leveraging sparse, event-driven computations triggered only when neurons reach a threshold. This spike-based mechanism allows SNNs to achieve substantial energy savings, making them a powerful choice for neuromorphic applications. SNNs have shown success in vision tasks like image classification and object detection, through CNN-based and more recently, transformer-based architectures [5, 7, 11, 15, 17, 23, 24, 35, 44, 48–50].

However, applying SNNs to offline RL presents unique challenges. Traditional RL methods rely on precise estimation of state-action values or continuous policy functions, which are challenging to match with SNNs' discrete spike-based processing. As a result, most approaches focus on ANN-to-SNN conversion or hybrid frameworks combining ANNs and SNNs [29, 38]. Recently, offline RL has been approached as a CSM problem [3], framing it as a sequence prediction task to generate actions with a goal-conditioned policy. Transformers [40], known for their ability to model time-dependent features and manage large-scale sequential data, are widely used to directly model trajectories in offline RL. While spike-driven transformers have proven effective in vision tasks, their application in sequential decision-making tasks with fine-grained temporal dynamics presents additional complexities.

In this work, we introduce Decision SpikeFormer (DSFormer), the first spike-driven transformer model for offline RL. Unlike SNN transformers in vision, which focus on spatial dependencies, DSFormer's self-attention de-

*This work was done when Wei Huang interned at Shanghai Artificial Intelligence Laboratory.
†Qinying Gu and Nanyang Ye are co-corresponding authors.

sign emphasizes temporal and positional dependencies. We develop Temporal Spiking Self-Attention (TSSA), which concatenates inputs along the temporal dimension before applying self-attention to enhance global temporal dependencies—essential for effective credit assignment over long sequences. To capture local dependencies and improve energy efficiency, we further introduce Positional Spiking Self-Attention (PSSA) with a learnable positional bias and element-wise computations. The localized window in PSSA reduces computational complexity, aligning well with the efficiency and speed requirements of high-dimensional SNNs. Additionally, we propose Progressive Threshold-dependent Batch Normalization (PTBN) to address the disruptive effects of LayerNorm on SNN spiking behavior. PTBN transitions from LayerNorm to SNN-compatible BatchNorm during training and simplifies to BatchNorm at inference, preserving both temporal dependencies and spiking characteristics.

We summarize our contributions as follows:

- We develop two new self-attention mechanisms, TSSA and PSSA, designed for sequence modeling tasks in offline RL, tailored to capture temporal and positional dependencies, distinguishing them from prior spike-driven self-attention focused on vision tasks.
- We introduce PTBN, a progressive normalization method combining the advantages of LayerNorm and BatchNorm that preserve the temporal dependency and maintain the spiking nature of SNNs in the mean time.
- We validate DSFormer on D4RL, achieving 78.4% energy savings and surpassing SNN and even ANN counterparts in performance, showing its potential for both energy efficiency and superior task results, marking a significant advancement in SNN applications.

2. Related Work

2.1. Offline Reinforcement Learning

In offline RL, early approaches such as behavior cloning (BC) [39] mapped states to actions from demonstrations but were limited by data diversity and distribution mismatch. Policy conservatism and regularization [14, 18, 26, 28] addressed these challenges by constraining learning within the dataset's support, improving stability and reducing overestimation. Recent advances have redefined the task as CSM [1, 6, 12, 16], frames offline RL as a supervised sequence prediction task. This approach, exemplified by Decision Transformer (DT) [3], uses transformers to capture long-term dependencies by conditioning actions on past states and future returns, eliminating the need for bootstrapping. The simplicity of the DT structure inspires many follow-up work [2, 31, 33, 42] using Transformer [40] to model the temporal features across diverse decision-making and control tasks. Recently, Some works pointed out DT's limitation of stitching optimal sequences from suboptimal data, and they address this issue by integrating value-based regularization (e.g. Q-value) [13, 41] to help balance trajectory modeling with optimal action selection. In this work, we focus on designing an SNN model within the DT framework without adding any extra value-based regularization.

2.2. Spike-driven Transformers

Vision Tasks Spikformer [50] pioneered spiking vision transformers, introducing spiking self-attention to eliminate traditional multiplications and softmax, using spiking neurons and BatchNorm in place of LayerNorm and GELU. Spikingformer [48] proposed a spike-driven residual learning to avoid non-spike computations in Spikformer. SpikeFormer [43] and Meta-SpikeFormer [44] developed four spike-driven self attentions, making SNN transformers applicable across tasks like classification and segmentation. SpikingResformer [32] also showed competitive vision task results with a ResNet-inspired, dual self-attention design.

Sequence Modeling Tasks Spike-driven transformers have also advanced in sequence modeling for NLP tasks. SpikeGPT [51], for example, replaces standard self-attention with an element-wise spiking RWKV module to maintain temporal dependencies, while SpikeBERT [25] directly adapts Spikformer to the BERT architecture. Similarly, SNN-BERT [36] introduces bidirectional spiking neurons to enhance temporal modeling capabilities in NLP. However, despite these advances, many of these models still rely on floating-point calculations, particularly in softmax and LayerNorm, during training and inference. More detailed Related Work is presented in **Appendix B**.

3. Method

3.1. Preliminary

Spiking Neural Network In the spiking neuron layer, we adopt the Leaky Integrate-and-Fire (LIF) model, known for its ability to mimic biological neuron behavior and efficiently handle temporal information [27]. The dynamics of the LIF neuron are described by the following equation:

$$\begin{aligned} U^t &= H^{t-1} + I^t \\ S^t &= \mathrm{Hea}\left(U^t - U_{th}\right) \\ H^t &= U_{\text{reset}}\, S^t + \gamma U^t \left(1 - S^t\right) \end{aligned} \tag{1}$$

where I^t is the input to the LIF neuron at time step t, U^t is the membrane potential that integrate I^t and temporal input H^{t-1}. $\mathrm{Hea}(\cdot)$ is a Heaviside function which equals 1 for $x \geq 0$ and 0 otherwise. When U^t exceeds the firing threshold U_{th}, the neuron fires a spike signal S^t and the membrane potential is reset to U_{reset} ; otherwise, the membrane potential decays to $H^t = \gamma U^t$, where $\gamma \leq 1$ is the decay factor of the membrane potential. For simplicity, we denotes the spiking neuron layer as $\mathcal{SN}(\cdot)$ in subsequent sections.

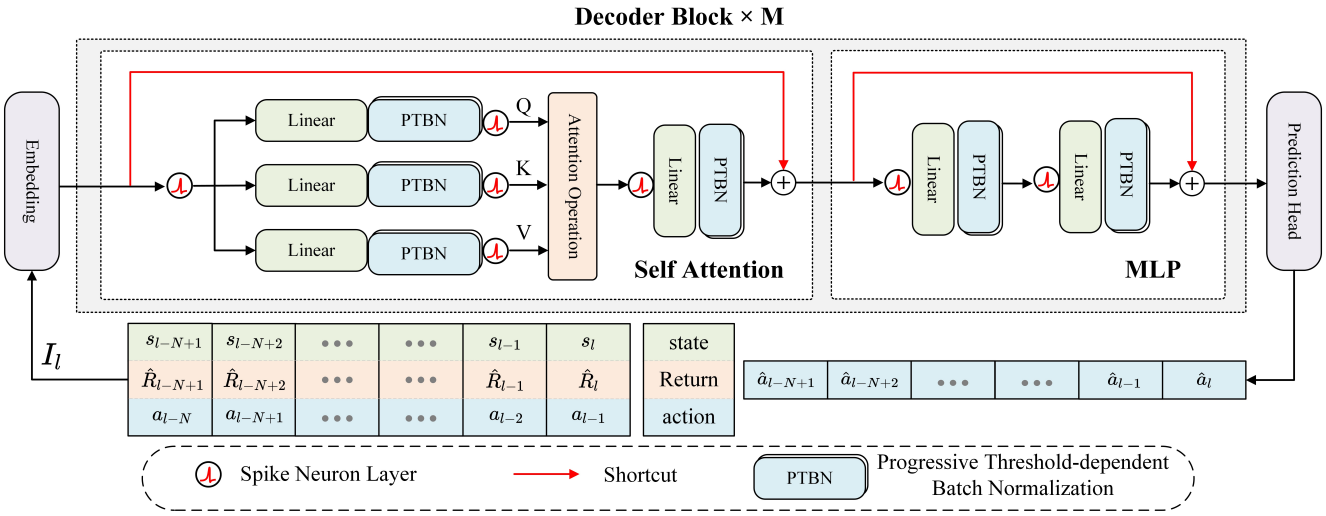


Figure 1. The overall architecture of DSFormer. The input sequence I_l is embedded, repeated T times, fed into the Decoder Blocks through a spike-driven self-attention and MLP layer in each block, and finally passed to the Prediction Head to generate next action predictions.

Offline RL The goal of RL is to learn a policy maximizing the expected cumulative return $\mathbb{E}[R(\tau)]$ in a Markov Decision Process (MDP) which is formulated by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, including states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $P(s'|s,a)$, and a reward function $r = \mathcal{R}(s,a)$, where τ denotes the trajectory sequence. In offline RL, algorithms learns a policy π_θ entirely from a static dataset $\mathcal{D} = \{\tau | \tau \sim \pi_\beta, \tau = (s_0, a_0, r_0, \ldots, s_L, a_L, r_L)$ collected by an unknown behavior policy π_β. To generate actions based on future desired rewards, we adopt the method to feed return-to-go (i.e., the suffix of the reward sequence, $\hat{R}_l = \sum_{l'=l}^{L} r_l$) into the trajectory τ instead of feeding the rewards directly [3].

3.2. Overall Architecture

The DSFormer, inspired by the DT, employs an autoregressive Transformer architecture, as illustrated in Figure 1. The network consists of an Embedding Layer, M stacked Decoder Blocks, and a Prediction Head. Each Decoder Block follows standard SNN Transformer designs [44, 50], incorporating spike-driven Self-Attention and a spike-driven Multi-Layer Perceptron (MLP). In the Self-Attention layer, two spiking self-attention mechanisms, TSSA and PSSA are proposed considering the Markov nature in offline RL. PTBN is proposed and used instead of BatchNorm to accommodate the autoregressive sequence decision-making process in SNNs, and Membrane Shortcut (MS) [15] is adopted in the achitecture. Both the Embedding Layer and the Prediction Head are implemented through fully connected layers.

At each iteration step l for the agent to interact with the environment, the input sequence is formalized as: $I_l = (a_{l-N}, \hat{R}_{l-N+1}, s_{l-N+1}, \ldots, a_{l-1}, \hat{R}_l, s_l)$, where N represents the context length of historical information for making a decision. a, $\hat{R}$, and s represent the action, estimated return-to-go, and state, respectively. Note that for clarity, t denotes the inner time step of the SNN, while l refers to the iteration step in offline RL. Each element in I_l is concatenated and embedded to form the input matrix as: $X_l = \{\mathrm{Emb}(a_{l-N}, \hat{R}_{l-N+1}, s_{l-N+1}); \mathrm{Emb}(a_{l-N+1}, \hat{R}_{l-N+2}, s_{l-N+2}); \ldots; \mathrm{Emb}(a_{l-1}, \hat{R}_l, s_l)\} \in \mathbb{R}^{N \times D}$, where D is the channel dimension, and the number of tokens equals the context length N. X_l is then repeated T times along the temporal dimension (T being the total SNN timesteps) and fed into the Decoder Blocks. For a comprehensive notation table, see **Appendix A**.

In each Decoder Block, the input X passes through a spike-driven Self-Attention layer to yield $Y = X + \text{Self-Attention}(X)$, followed by an MLP layer to produce the final output $Z = Y + \text{MLP}(Y)$. The output from the last Decoder Block is normalized along the temporal dimension and passed to the Prediction Head to generates the next action predictions.

3.3. Spike-driven Self-attention

Vanilla Self-Attention As shown in Figure 2(a), the vanilla self-attention (VLA) can be formulated as follows. For clarity, we will explain within a single attention head in the following discussions:

$$Q = XW_q, K = XW_k, V = XW_v$$
$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad (2)$$

where Q, K and V are the query, key, and value matrices, obtained by applying three learnable linear transformations

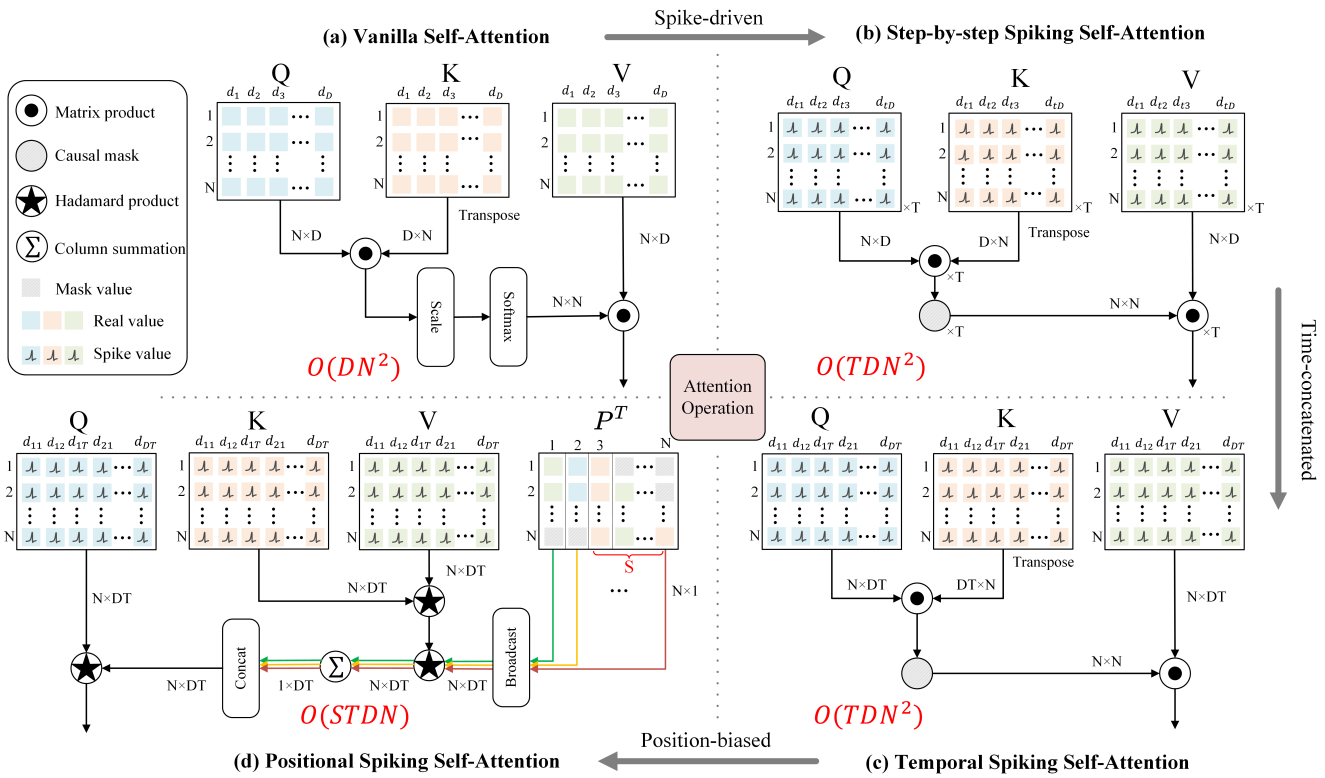


Figure 2. Self-attention mechanisms with different computational complexity. (a) VSA inherited from [40]. (b) SSSA, a spike version of VSA. (c) TSSA that concatenates inputs across the temporal dimension before self-attention. (d) PSSA that incorporates positional bias. To simplify the plotting, we set T = 3.

to the original input X, with W_q, W_k and W_v as the corresponding weight matrices. d_k is the channel dimension of K. The VLA mechanism has a high quadratic complexity of $O(DN^2)$, with respect to the token numbers. Besides, the Softmax and division operations are not compatible with the addition nature of SNNs.

Step-by-step Spiking Self-Attention To address this, we first propose the Step-by-Step Spiking Self-Attention (SSSA) shown in Figure 2(b), which is inspired by the design of the spiking self attention (SSA) [50] but with a causal masking. Intuitively, the causal masking prevents predicting the current action with the next state, action, and return-to-go tuple. SSSA repeats self-attention computations at each time step. As shown in Equation 3, the spike matrices Q, K, V are first generated by mapping the input X with three weight matrices and then normalizing them with the proposed PTBN, which will be explained later.

$$Q, K, V = \mathcal{SN}\left(\text{PTBN}\left(XW_q\right)\right), \mathcal{SN}\left(\text{PTBN}\left(XW_k\right)\right), \mathcal{SN}\left(\text{PTBN}\left(XW_v\right)\right) \tag{3}$$

where W_q, W_k and W_v are the linear transformation matrices for Q, K and V, and $\mathcal{SN}(\cdot)$ is the spiking neuron layer.

The SSSA computation at each time step is given by:

$$\text{Attention}\left(Q^t, K^t, V^t\right) = \text{mask}\left(Q^t (K^t)^\top\right) V^t \tag{4}$$

where Q^t, K^t and V^t correspond to the input of Q, K and V at t-th time step, respectively. A causal mask is applied to prevent the model from attending to future information. The overall time complexity for SSSA is $O(TDN^2)$, with its calculation presented in **Appendix C**

Temporal Spiking Self-Attention As shown in Figure 2 (c), Temporal Spiking Self-Attention (TSSA) is then introduced to address the limitation of SSSA. Due to its single-step property, SSSA cannot effectively exploit the temporal nature of SNNs to better capture dependencies within the (actions, states, rewards-to-go) tuples. TSSA overcomes this problem by concatenating the input across the temporal dimension before performing self-attention. This allows the model to weigh the dependencies over the entire sequence simultaneously. The TSSA computation is formulated as:

$$\text{Attention}(Q, K, V) = \text{mask}\left(QK^\top\right) V \tag{5}$$

where Q, K, V are concatenated Q^t, K^t, V^t respectively. This distinguishes the TSSA from previous spike-driven attentions used for image classification tasks, which do not

involve complex temporal structures. We further prove that the temporal-concatenated inputs can capture essential information than the non-concatenated version by reducing the entropy. We prove this from an information-theoretic perspective:

Theorem 1 *Let X^t represent the input at time step t, then the joint entropy after concatenation satisfies: $H\left(X^1, X^2, \ldots, X^T\right) = H\left(X^1\right) + H\left(X^2 \mid X^1\right) + \ldots H\left(X^T \mid X^1, X^2, \ldots, X^{T-1}\right)$. Since X^t and X^{t+1} are not independent according to the LIF dynamics: $H\left(X^{t+1} \mid X^1, X^2, \ldots, X^t\right) < H\left(X^{t+1}\right)$. Therefore, we can conclude: $H\left(X^1, X^2, \ldots, X^T\right) < H\left(X^1\right) + H\left(X^2\right) + \ldots + H\left(X^T\right)$.*

Remark: The Theorem indicates that TSSA reduces input entropy, facilitating more effective pattern learning. Its time complexity remains $O(TDN^2)$, comparable to SSSA, without added computational cost.

Positional Spiking Self-Attention While TSSA enhances the modeling of temporal properties in offline RL, it has limitations: (1) it lacks local-dependency modeling and (2) has a quadratic complexity with respect to the token number N. To address these issues, motivated by [34, 46], we propose Positional Spiking Self-Attention (PSSA), which captures local dependencies more effectively in offline RL. Unlike SSSA and TSSA that implement spiking versions of VLA without considering the locality inherent to RL trajectories modeled as Markov decision processes (MDP), PSSA introduces a learnable pair-wise positional bias to model these local associations. This design not only improves modeling accuracy but also enhances speed and reduces energy consumption. As illustrated in Figure 2 (d), PSSA incorporates spiking mechanisms, temporal concatenation, and learnable pair-wise positional biases. The PSSA computation is:

$$\text{Attention}(Q, K, V)_i = Q_i \odot \sum_{j=1}^{N} P_{ij} \odot K_j \odot V_j \quad (6)$$

where $\odot$ denotes element-wise multiplication, and Q_i, K_j and V_j represent the feature vectors corresponding to the i-th query and j-th key-value tokens, respectively. The matrix $P \in \mathbb{R}^{N \times N}$ consists of learnable pair-wise positional biases, where each element P_{ij} captures the positional relationship between the i-th query token and the j-th key-value token pair. This formulation first combines keys and values to capture token-to-token dependencies, then modulates these interactions with positional biases, and finally integrates with the query vector to produce context-aware information for each token. To account for the local dependencies of offline RL tasks, we use the following learnable relative positional bias:

$$P_{ij} = \begin{cases} P_{ij}, & \text{if } |i-j| < S \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

where S is the local window size. The time complexity of PSSA is $O(STDN)$; since S is a constant with $S \ll N, D$, the time complexity can be approximated as $O(TDN)$.

3.4. Progressive Threshold-dependent Batch Normalization (PTBN)

Threshold-dependent batch normalization (tdBN) [47] is widely used and essential in deep SNNs to mitigate gradient explosion and vanishing. However, in sequence modeling for offline RL, preserving dependencies within each sequence is essential, as each iteration step relies on context from surrounding steps. tdBN, which normalizes across the batch dimension, may disrupt these dependencies by introducing unintended interactions between sequences. In contrast, LayerNorm is more appropriate as it can normalize each time and iteration step independently, preserving temporal dependencies and ensuring stability across variable-length sequences. The combination of Linear layer and LayerNorm is widely adopted in standard Transformer where LayerNorm normalizes inputs along the feature dimension without accounting for the sequence length. To adapt this for SNN sequence modeling, we first propose Threshold-dependent Layer Normalization (tdLN) as shown in Figure 3.

Specifically, the input of the tdLN is denoted as $X \in \mathbb{R}^{B \times N \times T \times D}$, where B is the batchsize. A threshold voltage U_{th} and a scaling hyperparameter α are introduced to control the magnitude of normalized values, making them compatible with the threshold mechanism in SNNs. The tdLN is performed as:

$$\hat{X}_{ijkg} = \frac{\alpha U_{\text{th}}\left(X_{ijkg} - \mu_{ijk}\right)}{\sqrt{\sigma_{ijk}^2 + \epsilon}}, \quad Y_{ijkg} = \lambda \hat{X}_{ijkg} + \beta \quad (8)$$

where ϵ is a small positive constant to prevent division by zero, μ_{ijk} and σ_{ijk}^2 are the estimated mean and variance of X across the channel dimension D. We do not normalize over the context dimension N as in offline RL, the state should change drastically due to actions. λ and β are learnable parameters.

However, since μ_{ijk} and σ_{ijk}^2 vary for each batch of input (i.e. $\mu_{ijk}, \sigma_{ijk}^2 \in \mathbb{R}^{B \times N \times T}$), they are incompatible with the fixed weights $W \in \mathbb{R}^{D \times D'}$ and biases $b \in \mathbb{R}^{D'}$ of the Linear layer. This prevents merging the Linear layer and tdLN during inference, resulting in additional floating-point operations that disrupt the spiking nature of SNNs. To resolve this, motivated by [10], we propose Progressive

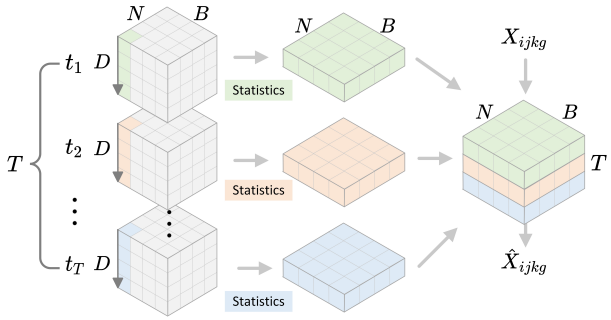


Figure 3. Design of tdLN. Each cube represents the feature map at timestep t and calculates statistics along the D dimension to obtain mean and variance with a shape of $B \times N \times T$ for normalization.

Threshold-dependent Batch Normalization (PTBN), which gradually transitions from tdLN to a SNN-compatible tdBN during training. According to [47], tdBN here is performed as $\frac{\alpha U_{\text{th}}(X_{ijkg}-\mu_g)}{\sqrt{\sigma_g^2+\epsilon}}$. PTBN combines tdLN and tdBN using a gradually updated weight:

$$\text{PTBN}(x) = \theta\,\text{tdLN}(x) + (1-\theta)\,\text{tdBN}(x) \tag{9}$$

The parameter θ controls the balance between tdLN and the tdBN, fine-tuning the adjustment during training. Initially, θ starts at 1, favoring tdLN, and linearly decreases to tdBN as training progresses:

$$\theta = \frac{T_p - T_{cur}}{T_p}, \theta \in [0,1] \tag{10}$$

where T_p is the predetermined training step for PTBN, and T_{cur} is the current training step. During training, T_p is allocated a predefined proportion of total steps, with the remainder dedicated to tdBN for BatchNorm adaptation. During inference, PTBN reduces to the tdBN form, which can be merged with fully connected layers, eliminating extra computations and preserving the spiking characteristics of SNNs. Further details on the calculation and implementation can be found in the **Appendix D**.

4. Experiments

4.1. Experiment Setting

In this section, we present a comprehensive evaluation of DSFormer using the widely recognized D4RL benchmark [8]. A broad range of ANN and SNN baselines are included. Ablation studies are also conducted to examine the individual contributions of key components. We further discuss why DSFormer's SNN structure is well-suited for offline RL tasks, emphasizing its advantages in handling temporal dependencies and energy-efficient learning.

Baseline We begin with classic MLP-based and transformer-based ANN methods, including Behavior Cloning (BC) [39], Conservative Q-Learning (CQL) [20], Decision Transformer (DT) [3], and Fourier Controller Network (FCNet) [37]. For SNN architectures, since no specific SNN models are designed for offline RL, we select SNN models from NLP, including SpikeGPT [51] and SpikeBERT [25], considering the sequence modeling nature of our work.

Datasets and settings We evaluate the DSFormer on the MuJoCo and Adroit environments, which encompass a variety of complex control tasks. The MuJoCo environments—such as HalfCheetah, Hopper, and Walker2d—cover various locomotion modes with continuous action spaces and dense rewards. The Adroit datasets, based on noisy human demonstrations, include tasks like Door, Hammer, and Relocate, which emphasize precise manipulation in high-dimensional spaces. All models are trained on these collected datasets and evaluated in simulators. We report normalized scores following the protocol in [8], where a score of 100 represents expert-level performance, and a score of 0 reflects random agent performance.

Number of parameters To ensure a fair comparison between DSFormer and its ANN counterpart, DT, both models are based on the same MetaFormer [45] architecture, sharing almost identical parameter counts. We apply PTBN before each spiking neuron in DSFormer to prevent gradient explosion during training. While PTBN adds extra parameters during training, it is merged into the Linear layer at inference, resulting in nearly the same parameter counts for DSFomer and DT (0.3% fewer for TSSA and 0.08% more for PSSA, compared to DT). We set $T = 4$ for all tasks. Further details on experimental and implementation settings are provided in **Appendix E**.

4.2. Experiment results

Results for MuJoCo Domain The experimental results on the MuJoCo environment (Table 1) demonstrate the superior performance of DSFormer. Both spike attention mechanisms, TSSA and PSSA, not only outperform other transformer-based SNN methods by a significant margin but also achieve top average scores (75.7 and 78.8, respectively) in the ANN domain, with substantially improved energy efficiency (410.5 μJ in DT versus 96.1 μJ in TSSA and 88.8 μJ in PSSA). SNN architectures perform poorly because: SpikeBERT's shortcut connection introduces integer values that disrupt residual propagation, compromising spike characteristics, while SpikeGPT, designed for NLP, struggles with the limited data in offline RL. TSSA and PSSA, however, consistently rank first or second across all subtasks. Additionally, PSSA excels in most tasks, underscoring its ability to efficiently capture local dependencies in the motion-control MDP due to the introduction of pair-wise

Table 1. **Results on MuJoCo.** We report mean and variance scores from five random seeds, following [8]. Dataset abbreviations: 'medium' as 'm' (medium policy achieving one-third of expert score); 'medium-replay' as 'm-r' (mixed-quality trajectories from training); 'medium-expert' as 'm-e' (trajectories combining medium and expert policies). The strengthened digits denote the highest scores.

MuJoCo Tasks	BC	CQL	DT	FCNet	SpikeGPT	SpikeBert	TSSA	PSSA
halfcheetah-m-e	35.8	62.4	86.8±1.3	91.2±0.3	23.6±4.5	24.3±6.0	91.3±0.2	**91.5±0.3**
walker2d-m-e	6.4	98.7	108.1±0.2	108.8±0.1	22.6±4.8	92.5±22.4	108.6±0.2	**108.9±0.1**
hopper-m-e	**111.9**	111.0	107.6±1.8	110.5±0.5	32.7±5.4	84.1±8.8	111.0±0.7	110.9±0.2
halfcheetah-m	36.1	44.4	42.6±0.1	**42.9±0.4**	26.9±0.8	20.0±3.5	42.5±0.4	42.8±0.3
walker2d-m	6.6	79.2	74.0±1.4	**75.2±0.5**	16.4±10.2	22.9±10.4	72.4±4.5	**75.2±1.4**
hopper-m	29.0	58.0	67.6±1.0	57.8±6.0	25.1±6.4	31.4±4.9	64.6±2.1	**74.1±4.3**
halfcheetah-m-r	38.4	46.2	36.6 ±0.8	**39.8±0.8**	21.8±2.0	32.2±8.0	38.7±1.1	38.8±0.7
walker2d-m-r	11.3	26.7	66.6±3.0	63.5±7.5	16.7±3.3	21.2±6.4	66.0±3.3	**71.0±3.6**
hopper-m-r	11.8	48.6	82.7±7.0	85.8±1.7	51.5±7.1	30.1±8.6	85.8±3.6	**96.3±1.3**
MuJoCo mean ↑	31.9	63.9	74.7	75.1	26.4	39.9	75.7	**78.8**
Power (μJ) ↓	N/A	N/A	410.5	1022.03	**27.8**	806.7	96.1	88.8

Table 2. **Results on Adroit.** Mean and variance scores from five random seeds, following [8]. Abbreviations: 'expert' as 'm-e' (expert policy); 'human' as 'h' (human demonstrations); and 'cloned' as 'c' (behavior cloning). The strengthened digits denote the highest scores.

Adroit Tasks	BC	CQL	DT	FCNet	SpikeGPT	SpikeBert	TSSA	PSSA
pen-e	85.1	107.0	110.4±20.9	108.0±11.3	30.5±10.3	46.2±19.5	104.6±13.2	**122.0±17.8**
door-e	34.9	101.5	95.5±5.7	102.9±2.9	65.3±16.9	96.4±4.6	105.0±0.3	**105.2±0.1**
hammer-e	125.6	86.7	89.7±24.6	121.1±6.1	51.1±18.7	71.3±16.5	126.4±0.4	**127.2±0.3**
relocate-e	101.3	95.0	15.3±3.6	50.0±6.0	0.7±0.9	0.3±0.5	106.3±2.6	**108.4±2.2**
pen-h	34.4	37.5	-0.2±1.8	57.7±11.1	29.8±11.7	20.0±16.7	**89.7±10.0**	75.7±25.1
door-h	0.5	**9.9**	0.1±0.0	0.4±0.5	0.1±0.0	0.2±0.0	0.4±0.1	0.2±0.0
hammer-h	1.5	**4.4**	0.3±0.0	1.2±0.0	0.3±0.0	0.3±0.0	0.4±0.1	0.2±0.0
relocate-h	0.0	0.2	**0.2±0.2**	0.0±0.0	0.1±0.0	0.0±0.0	0.0±0.0	0.0±0.0
pen-c	**56.9**	39.2	22.7±17.1	50.4±24.1	17.0±22.0	17.6±29.0	41.1±19.7	44.8±14.7
door-c	-0.1	**0.4**	0.1±0.0	-0.2±0.0	0.2±0.0	0.2±0.0	**0.4±0.8**	0.0±0.0
hammer-c	0.8	**2.1**	0.3±0.0	0.2±0.0	0.3±0.0	0.3±0.5	0.2±0.0	0.2±0.0
relocate-c	-0.1	-0.1	-0.3±0.0	-0.2±0.0	**0.1±0.0**	0.0±0.5	-0.2±0.0	-0.2±0.0
Adroit Mean ↑	36.7	40.3	27.8	41.0	16.3	21.1	47.9	**48.6**

positional bias. Further analysis is provided in the Ablation study. Moreover, PSSA's element-wise multiplication and small local window size ($S = 8$) yield lower energy consumption (31.0 μJ) in Self-Attention mechanism than TSSA (44.6 μJ) and DT (168.2 μJ), as detailed in **Appendix F**.

Results for Adroit Domain Table 2 highlights the effectiveness of DSFormer, which significantly outperforms other ANN and SNN methods. Both TSSA and PSSA surpass DT across all sub-tasks, with an substantial performance gains (47.9 and 48.6 versus 27.8). DSFormer's advantages are more pronounced in Adroit compared to MuJoCo, as its temporal dynamics effectively capture the long-range, high-dimensional action dependencies and sparse rewards essential for precise manipulation tasks in Adroit, as discussed further in Section 4.4. Overall, DSFormer shows exceptional capability in managing complex motion patterns in an energy-efficient, low-power manner. Visualization results are presented in **Appendix Figure. S1**.

Table 3. Ablation Study on DSFormer

Ablation	Methods	Hopper-m-e	Hopper-m	Hopper-m-r	Average
Ours	TSSA	**111.0**	64.6	85.8	87.2
	PSSA	110.9	**74.1**	**96.3**	**93.8**
Spike self-attention	SDSA-1	104.3	54.8	47.1	68.7
	SDSA-3	61.5	56.1	30.7	49.4
Normalization Method	None	89.7	49.1	21.7	53.5
	tdBN	110.8	64.2	78.5	84.5
	tdLN	**111.0**	72.0	93.1	92.0

4.3. Ablation Study

We conduct ablation studies on spiking self-attention mechanisms and normalization method. More ablations including SNN timestep T and local window size in PSSA are presented in **Appendix G**.

Design of Spiking Self-Attention Mechanisms To assess the impact of spike-driven self-attention mechanisms

Table 4. Results on AntMaze with varied sequence length

AntMaze Mean	Length=50	Length=100	Length=150	Length=200
TSSA (ours)	**65**	**73**	**71**	**72**
DT	62	67	58	59

Table 5. Results on D4RL-Hopper datasets with sparse reward

Task Name	Dense Setting			**Sparse Setting**		
	CQL	DT	DSFormer	CQL	DT	DSFormer
Hopper-m-e	111.0	107.6	110.9	9.0	107.3	**110.7**
Hopper-m	58.0	67.6	74.1	5.2	60.7	**68.2**
Hopper-m-r	48.6	82.7	96.3	2.0	78.5	**88.9**

in offline RL tasks, we replaced our designs with those used in other SNN transformers, including SDSA-1 from [43] and SDSA-3 from [44], while keeping all other components in DSFormer unchanged. Validation was conducted in the MuJoCo-Hopper environment. The results in Table 3 show that both TSSA and PSSA consistently outperform alternative mechanisms by a substantial margin. The performance limitation of SDSA-1 arises from its simplistic computation, which restricts the model's feature extraction capacity, making it less effective in complex problem modeling. For SDSA-3, its poor performance primarily because it was designed for vision tasks that emphasize feature-dimension relationships rather than token-to-token dependencies. Given that offline RL is an autoregressive task requiring next-token prediction based on previous states, SDSA-3 is less effective in this context.

Normalization Methods To evaluate the effectiveness of Progressive Threshold-dependent Batch Normalization (PTBN), we compared it against using no normalization, the commonly used tdBN in SNNs, and tdLN—a LayerNorm variant we proposed for sequence modeling. These experiments were conducted on DSFormer with PSSA in the MuJoCo-Hopper environment. As shown in Table 3, performance significantly degrades without a Norm module, underscoring its importance in preventing gradient explosion and vanishing. PTBN achieves performance comparable to tdLN, while tdBN falls behind both. As discussed, tdBN is less effective for sequence modeling, and while tdLN offers stability, it disrupts SNN spiking properties. PTBN effectively combines the advantages of LayerNorm and BatchNorm, preserving temporal dependencies while maintaining the spiking nature essential to SNNs.

4.4. Discussion

In this section, we investigate the reasons of why DSFormer is effective in offline RL from the perspectives of long-range temporal dependencies and sparse reward processing.

Long-range Temporal Dependencies We evaluated DSFormer on tasks with substantial long-range temporal dependencies, specifically the AntMaze environment, which involves sparse rewards and complex maze navigation using an 8-DoF "Ant" quadruped robot in a multi-goal, non-Markovian setting. To compare DSFormer with DT, we tested sequence lengths of 50, 100, 150, and 200 steps. As shown in Table 4, DSFormer consistently outperforms DT, with less performance degradation as sequence length increases. This advantage arises from the threshold-based activation in SNN neurons, which accumulate membrane potential and fire only when a threshold is met, then reset. This mechanism allows SNNs to remain inactive between critical events, preserving information over extended intervals without continuous updates, thus enhancing sensitivity to prior states and improving performance in tasks with significant temporal gaps.

Sparse Reward Processing To assess DSFormer's advantage in sparse reward settings, we compared the performance of DSFormer, DT and CQL under sparse (delayed) reward conditions in the MuJoCo-Hopper tasks, where rewards are granted only at the final timestep of each trajectory. Table 5 demonstrates that delayed returns minimally affect DSFormer and DT while CQL collapses. This resilience is due to SNNs' event-driven nature, which triggers neuron activation only in response to key events, such as critical state changes or sparse reward signals, while remaining inactive during less relevant phases. This selective activation allows SNNs to focus on essential information, reducing interference from irrelevant data and enhancing responsiveness to reward-related events.

5. Conclusion

DSFormer is the first spike-driven transformer for offline RL via sequence modeling, employing novel self-attention mechanisms TSSA and PSSA to capture temporal and positional dependencies and PTBN normalization to replace SNN-incompatible LayerNorm while preserving temporal dependencies. The threshold-based activation in SNN neurons enables DSFormer to handle long-range dependencies and sparse environments effectively. Results on the D4RL benchmark indicate that DSFormer outperforms both SNN and ANN counterparts while achieving 78.4% energy savings, demonstrating strong potential for energy-efficient embodied applications. A current limitation is the lack of neuromorphic chip deployment, which will be the focus of future work.

Acknowledgments This work is supported by Shanghai Artificial Intelligence Laboratory and the National Natural Science Foundation of China under Grant No.62106139.

References

[1] David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, 35:1542–1553, 2022. 2

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 2

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 1, 2, 3, 6

[4] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018. 1

[5] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022. 1

[6] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021. 2

[7] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021. 1

[8] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 6, 7

[9] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019. 1

[10] Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. Slab: Efficient transformers with simplified linear attention and progressive re-parameterized batch normalization. In *International Conference on Machine Learning*, 2024. 5

[11] Yufei Guo, Xiaode Liu, Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Rmploss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17391–17401, 2023. 1

[12] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Prompt-tuning decision transformer with preference ranking. *arXiv preprint arXiv:2305.09648*, 2023. 2

[13] Shengchao Hu, Ziqing Fan, Chaoqin Huang, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Q-value regularized transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.17098*, 2024. 2

[14] Xiaohan Hu, Yi Ma, Chenjun Xiao, Yan Zheng, and HAO Jianye. Iteratively refined behavior regularization for offline reinforcement learning. 2023. 2

[15] Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. 1, 3

[16] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021. 1, 2

[17] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11270–11277, 2020. 1

[18] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021. 1, 2

[19] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019. 1

[20] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. 1, 6

[21] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. 1

[22] Guoqi Li, Lei Deng, Huajin Tang, Gang Pan, Yonghong Tian, Kaushik Roy, and Wolfgang Maass. Brain-inspired computing: A systematic survey and future trends. *Proceedings of the IEEE*, 2024. 1

[23] Yang Li, Xiang He, Yiting Dong, Qingqun Kong, and Yi Zeng. Spike calibration: Fast and accurate conversion of spiking neural network for object detection and segmentation. *arXiv preprint arXiv:2207.02702*, 2022. 1

[24] Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. *arXiv preprint arXiv:2407.20708*, 2024. 1

[25] Changze Lv, Tianlong Li, Jianhan Xu, Chenxi Gu, Zixuan Ling, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Spikebert: A language spikformer learned from bert with knowledge distillation. 2, 6

[26] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 1711–1724, 2022. 2

[27] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10 (9):1659–1671, 1997. 1, 2

[28] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017. 2

[29] Devdhar Patel, Hananel Hazan, Daniel J Saunders, Hava T Siegelmann, and Robert Kozma. Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game. *Neural Networks*, 120:108–115, 2019. 1

[30] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019. 1

[31] Jinghuan Shang, Kumara Kahatapitiya, Xiang Li, and Michael S Ryoo. Starformer: Transformer with state-action-reward representations for visual reinforcement learning. In *European conference on computer vision*, pages 462–479. Springer, 2022. 2

[32] Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5610–5619, 2024. 2

[33] Max Siebenborn, Boris Belousov, Junning Huang, and Jan Peters. How crucial is transformer in decision transformer? *arXiv preprint arXiv:2211.14655*, 2022. 2

[34] Xiaotian Song, Andy Song, Rong Xiao, and Yanan Sun. One-step spiking transformer with a linear complexity. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3142–3150. International Joint Conferences on Artificial Intelligence Organization, 2024. Main Track. 5

[35] Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6555–6565, 2023. 1

[36] Qiaoyi Su, Shijie Mei, Xingrun Xing, Man Yao, Jiajun Zhang, Bo Xu, and Guoqi Li. Snn-bert: Training-efficient spiking neural networks for energy-efficient bert. *Neural Networks*, 180:106630, 2024. 2

[37] Hengkai Tan, Songming Liu, Kai Ma, Chengyang Ying, Xingxing Zhang, Hang Su, and Jun Zhu. Fourier controller networks for real-time decision-making in embodied learning. In *Forty-first International Conference on Machine Learning*, 2024. 6

[38] Weihao Tan, Devdhar Patel, and Robert Kozma. Strategy and benchmark for converting deep q-networks to event-driven spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 9816–9824, 2021. 1

[39] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018. 2, 6

[40] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1, 2, 4

[41] Yuanfu Wang, Chao Yang, Ying Wen, Yu Liu, and Yu Qiao. Critic-guided decision transformer for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 15706–15714, 2024. 2

[42] Muning Wen, Runji Lin, Hanjing Wang, Yaodong Yang, Ying Wen, Luo Mai, Jun Wang, Haifeng Zhang, and Weinan Zhang. Large sequence models for sequential decision-making: a survey. *Frontiers of Computer Science*, 17(6): 176349, 2023. 2

[43] Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo XU, and Guoqi Li. Spike-driven transformer. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2, 8

[44] Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo XU, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2, 3, 8

[45] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 6

[46] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021. 5

[47] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11062–11070, 2021. 5, 6

[48] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, 2023. 1, 2

[49] Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Qkformer: Hierarchical spiking transformer using qk attention. *arXiv preprint arXiv:2403.16552*, 2024.

[50] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 1, 2, 3, 4

[51] Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason Eshraghian. SpikeGPT: Generative pre-trained language model with spiking neural networks. *Transactions on Machine Learning Research*, 2024. 2, 6