
Supplementary Material for FLAME: Fast Long-context Adaptive Memory for Event-based Vision

Anonymous Author(s)

Affiliation

Address

email

1 A Supplementary Section A: Detailed Proofs

2 A.1 Computational Complexity of Event-Driven SSMs

3 **Lemma 1.** *Let the event-driven state-space model be governed by:*

$$\dot{x}(t) = Ax(t) + BS(t),$$

4 *where $x(t) \in \mathbb{R}^N$ is the internal state, $A \in \mathbb{R}^{N \times N}$ is the state transition matrix, and $S(t) \in \mathbb{R}^M$ is the*
5 *input event train. The computational complexity of updating the internal state $x(t)$ at each event is*
6 *$O(N^2)$.*

7 *Proof.* The event-driven state-space model is governed by:

$$\dot{x}(t) = Ax(t) + BS(t),$$

8 where $x(t) \in \mathbb{R}^N$ represents the internal state of the system, $A \in \mathbb{R}^{N \times N}$ is the state transition matrix,
9 and $S(t) \in \mathbb{R}^M$ represents the input event train. When an event occurs at time t_i , the state update can
10 be represented by the following integral equation for $t \in [t_i, t_{i+1})$:

$$x(t_i^+) = e^{A\Delta t_i} x(t_i^-) + \int_{t_i^-}^{t_i^+} e^{A(t_i^+ - \tau)} BS(\tau) d\tau,$$

11 where:

- 12 • t_i^- and t_i^+ are the times just before and after the event at t_i ,
- 13 • $\Delta t_i = t_i^+ - t_i^-$ is infinitesimal,
- 14 • $S(\tau)$ contains Dirac delta functions at event times and is zero elsewhere.

15 For simplicity, we focus on the update at the event time t_i to approximate the state transition at each
16 event.

17 The update of the internal state $x(t)$ requires computing the matrix exponential $e^{A\Delta t}$, where $\Delta t = t - t_i$
18 represents the time interval between successive events. Computing the exact matrix exponential for a
19 general matrix $A \in \mathbb{R}^{N \times N}$ is computationally expensive, involving $O(N^3)$ operations using standard
20 algorithms such as diagonalization or the Schur decomposition.

21 To reduce the computational cost, we approximate the matrix exponential using a truncated Taylor
22 series expansion:

$$e^{A\Delta t_i} \approx I + A\Delta t_i + \frac{1}{2}A^2\Delta t_i^2.$$

where I is the identity matrix of size $N \times N$. This approximation is typically sufficient for small Δt , which is common between events.

In the Taylor series expansion approximation of $e^{A\Delta t}$, the dominant computational cost arises from multiplying the matrix $A \in \mathbb{R}^{N \times N}$ by itself and by the state vector $x(t) \in \mathbb{R}^N$.

The product $Ax(t)$, where $A \in \mathbb{R}^{N \times N}$ and $x(t) \in \mathbb{R}^N$, requires N^2 multiplications. Thus, the computational cost for this step is $O(N^2)$.

The term A^2 is computed by multiplying A by itself. Since A is an $N \times N$ matrix, computing A^2 explicitly would have a computational cost of $O(N^3)$. However, we avoid this by computing $A(Ax(t))$, which involves two sequential matrix-vector products, each costing $O(N^2)$. Therefore, the computational cost of computing $A^2x(t)$ is $O(N^2)$.

The term $BS(t)$, where $B \in \mathbb{R}^{N \times M}$ and $S(t) \in \mathbb{R}^M$, involves $O(NM)$ operations. Assuming M is proportional to N or smaller, this computation contributes $O(N^2)$ to the overall complexity.

To update the internal state $x(t)$, we perform the following operations: First, we multiply A by $x(t)$: $O(N^2)$; then multiply A^2 by $x(t)$: $O(N^2)$; followed by multiplying B by $S(t)$: $O(NM)$ and finally add the resulting vectors.

Thus, the overall computational complexity for updating the internal state $x(t)$ at each event is $O(N^2)$.

In the general case, where A is a dense matrix, the cost of updating the state is $O(N^2)$. If the matrix A has a specific structure, such as being sparse or block-diagonal, the computational cost can be reduced. For example: - If A is sparse with k non-zero entries per row, the cost of multiplying A by $x(t)$ becomes $O(kN)$, which can be significantly lower than $O(N^2)$ when $k \ll N$. - If A is block-diagonal, the cost can be reduced to $O(N)$ per block, depending on the number and size of the blocks. However, for the general case where no such structure is assumed, the computational complexity remains $O(N^2)$. The computational complexity of updating the internal state $x(t)$ at each event, using the matrix exponential approximation with a Taylor series expansion, is dominated by the matrix-vector multiplication operations. Additionally, accounting for the $BS(t)$ term maintains the overall complexity at $O(N^2)$. Therefore, the overall computational complexity for updating the internal state at each event is $O(N^2)$.

□

A.2 Long-context Temporal Dependency Preservation Via Event-Based Hippo

Theorem 1. Let $x(t) \in \mathbb{R}^N$ evolve according to

$$\dot{x}(t) = Ax(t) + BS(t),$$

where: - $A \in \mathbb{R}^{N \times N}$ is a HiPPO matrix with all eigenvalues satisfying $\text{Re}(\lambda_i) < 0$ for $i = 1, 2, \dots, N$,
- $B \in \mathbb{R}^{N \times M}$ is the input matrix, - $S(t) \in \mathbb{R}^M$ is the input event train, assumed to be bounded, i.e., there exists a constant $S_\infty > 0$ such that $\|S(t)\| \leq S_\infty$ for all $t \geq 0$, - $x_0 = x(0) \in \mathbb{R}^N$ is the initial state.

Then, the event-driven SSM preserves long-context temporal dependencies in the input event train $S(t)$, and the state $x(t)$ satisfies the bound:

$$\|x(t)\| \leq e^{-\alpha t} \|x_0\| + \frac{\|B\|S_\infty}{\alpha} (1 - e^{-\alpha t}),$$

where $\alpha = \min_i |\text{Re}(\lambda_i)| > 0$ is the memory retention factor determined by the eigenvalues of the HiPPO matrix A .

Proof. To establish the theorem, we will analyze the evolution of the internal state $x(t)$ governed by the differential equation:

$$\dot{x}(t) = Ax(t) + BS(t),$$

with initial condition $x(0) = x_0$.

The differential equation is a non-homogeneous linear ordinary differential equation (ODE). Using the variation of parameters method, the solution can be expressed as:

$$x(t) = e^{At} x_0 + \int_0^t e^{A(t-\tau)} BS(\tau) d\tau,$$

67 where: $-e^{At}x_0$ is the solution to the homogeneous equation $\dot{x}(t) = Ax(t)$ with initial condition
 68 $x(0) = x_0$, $-\int_0^t e^{A(t-\tau)}BS(\tau)d\tau$ accounts for the particular solution due to the input $S(t)$.
 69 Given that A is a HiPPO matrix, all its eigenvalues satisfy $\text{Re}(\lambda_i) < 0$ for $i = 1, 2, \dots, N$. This
 70 implies that A is a Hurwitz matrix, ensuring that the system is asymptotically stable. Define the
 71 memory retention factor α as:

$$\alpha = \min_i |\text{Re}(\lambda_i)| > 0.$$

72 This factor dictates the rate at which the influence of the initial state x_0 decays over time.

73 Consider the homogeneous solution $e^{At}x_0$. Since all eigenvalues of A have negative real parts, the
 74 matrix exponential e^{At} satisfies:

$$\|e^{At}\| \leq e^{-\alpha t},$$

75 where $\|\cdot\|$ denotes an operator norm (e.g., the induced 2-norm). This inequality leverages the spectral
 76 bound of A to provide an exponential decay rate.

77 Therefore, the contribution of the initial state is bounded by:

$$\|e^{At}x_0\| \leq \|e^{At}\| \cdot \|x_0\| \leq e^{-\alpha t}\|x_0\|.$$

78 Next, consider the particular solution:

$$\int_0^t e^{A(t-\tau)}BS(\tau)d\tau.$$

79 To bound its norm, apply the triangle inequality and properties of operator norms:

$$\left\| \int_0^t e^{A(t-\tau)}BS(\tau)d\tau \right\| \leq \int_0^t \|e^{A(t-\tau)}\| \cdot \|B\| \cdot \|S(\tau)\| d\tau.$$

80 Given that $\|S(\tau)\| \leq S_\infty$ and $\|e^{A(t-\tau)}\| \leq e^{-\alpha(t-\tau)}$, we have:

$$\left\| \int_0^t e^{A(t-\tau)}BS(\tau)d\tau \right\| \leq \|B\|S_\infty \int_0^t e^{-\alpha(t-\tau)} d\tau.$$

81 Evaluate the integral:

$$\int_0^t e^{-\alpha(t-\tau)} d\tau = \int_0^t e^{-\alpha s} ds = \frac{1 - e^{-\alpha t}}{\alpha}.$$

82 Thus, the bound becomes:

$$\left\| \int_0^t e^{A(t-\tau)}BS(\tau)d\tau \right\| \leq \frac{\|B\|S_\infty}{\alpha} (1 - e^{-\alpha t}).$$

83 Combining the bounds for the homogeneous and particular solutions, we obtain:

$$\|x(t)\| \leq \|e^{At}x_0\| + \left\| \int_0^t e^{A(t-\tau)}BS(\tau)d\tau \right\| \leq e^{-\alpha t}\|x_0\| + \frac{\|B\|S_\infty}{\alpha} (1 - e^{-\alpha t}).$$

84 This inequality demonstrates that the influence of the initial state x_0 decays exponentially at rate α .
 85 Also, the accumulated influence of the input event train $S(t)$ is bounded and grows to a steady-state
 86 value determined by $\|B\|$, S_∞ , and α .

87 The derived bound:

$$\|x(t)\| \leq e^{-\alpha t}\|x_0\| + \frac{\|B\|S_\infty}{\alpha} (1 - e^{-\alpha t}),$$

88 reveals that the term $e^{-\alpha t}\|x_0\|$ signifies that the system "forgets" its initial state exponentially fast,
 89 ensuring that old information does not dominate the state indefinitely. Also, the integral term captures
 90 the accumulated influence of the input event train $S(t)$. Since $S(t)$ is bounded, the state $x(t)$ can
 91 retain and reflect information from the input over extended periods without being overwhelmed by
 92 the initial condition.

93 Therefore, the event-driven SSM governed by a HiPPO matrix A effectively preserves long-context
 94 temporal dependencies in the input event train $S(t)$, while ensuring that the memory of the initial
 95 state x_0 decays at an exponential rate determined by α .

96 □

97 A.3 Error Bound For Event-Driven Matrix Exponential Approximation

98 **Lemma 2.** *Let the matrix exponential be approximated using a Taylor expansion up to the n -th term:*

$$e^{A\Delta t} \approx I + A\Delta t + \frac{A^2\Delta t^2}{2!} + \cdots + \frac{A^n\Delta t^n}{n!}.$$

99 Assume that the matrix norm $\|\cdot\|$ is submultiplicative, i.e., $\|AB\| \leq \|A\|\|B\|$ for any matrices A and
100 B of compatible dimensions. Then, the error E_n of this approximation satisfies

$$\|E_n\| \leq \frac{\|A\Delta t\|^{n+1}}{(n+1)!} e^{\|A\Delta t\|}.$$

101 *Proof.* The matrix exponential can be expressed as an infinite Taylor series:

$$e^{A\Delta t} = \sum_{k=0}^{\infty} \frac{(A\Delta t)^k}{k!}.$$

102 If we truncate this series after the n -th term, the remainder E_n is given by:

$$E_n = e^{A\Delta t} - \sum_{k=0}^n \frac{(A\Delta t)^k}{k!} = \sum_{k=n+1}^{\infty} \frac{(A\Delta t)^k}{k!}.$$

103 To bound the norm of the error E_n , we apply the submultiplicative property of the matrix norm:

$$\|E_n\| = \left\| \sum_{k=n+1}^{\infty} \frac{(A\Delta t)^k}{k!} \right\| \leq \sum_{k=n+1}^{\infty} \frac{\|A\Delta t\|^k}{k!}.$$

104 Using the submultiplicative property of the matrix norm:

$$\|E_n\| \leq \sum_{k=n+1}^{\infty} \frac{\|A\Delta t\|^k}{k!}.$$

105 Let $x = \|A\Delta t\| \geq 0$. Then:

$$\|E_n\| \leq \sum_{k=n+1}^{\infty} \frac{x^k}{k!}.$$

106 Since

$$\sum_{k=n+1}^{\infty} \frac{x^k}{k!} = e^x - \sum_{k=0}^n \frac{x^k}{k!} = R_n(x),$$

107 where $R_n(x)$ is the remainder of the Taylor series expansion of e^x .

108 According to Taylor's Remainder Theorem (Lagrange's form), there exists $\xi \in [0, x]$ such that:

$$R_n(x) = \frac{x^{n+1}}{(n+1)!} e^{\xi}.$$

109 Since $\xi \leq x$ and $e^{\xi} \leq e^x$ for $x \geq 0$, we have:

$$R_n(x) \leq \frac{x^{n+1}}{(n+1)!} e^x.$$

110 Therefore:

$$\|E_n\| \leq \frac{x^{n+1}}{(n+1)!} e^x = \frac{\|A\Delta t\|^{n+1}}{(n+1)!} e^{\|A\Delta t\|}.$$

Thus, the error E_n satisfies:

$$\|E_n\| \leq \frac{\|A\Delta t\|^{n+1}}{(n+1)!} e^{\|A\Delta t\|}.$$

□

—

A.4 Boundedness Of State Trajectories With Event Inputs

Theorem 2. Boundedness of State Trajectory in Event-Driven State-Space Models

For a given initial condition x_0 , the state trajectory $x(t)$ of the FLAME model driven by the event input $S(t)$ is bounded, i.e., $\|x(t)\| \leq C$, for some constant $C > 0$, provided that:

1. The input events $S(t)$ are of finite magnitude, i.e., $\|S(t)\| \leq S_\infty$ for all $t \geq 0$.
2. The decay matrix A_S is Hurwitz, meaning all its eigenvalues have negative real parts.
3. There exists a positive definite matrix P satisfying the Lyapunov equation $A_S^T P + P A_S = -Q$, for some positive definite matrix Q .

Proof. Consider the FLAME governed by:

$$\dot{x}(t) = A_S x(t) + B S(t),$$

where A_S is a Hurwitz matrix, B is the input matrix, and $S(t)$ is a bounded input event train with $\|S(t)\| \leq S_\infty$ for all $t \geq 0$.

We define a Lyapunov function $V(x) = x^T P x$, where P is a positive definite matrix satisfying the Lyapunov equation:

$$A_S^T P + P A_S = -Q,$$

with Q being a positive definite matrix. Such a P exists because A_S is Hurwitz. The derivative of $V(x)$ along the system trajectories is computed:

$$\dot{V}(x) = \frac{d}{dt}(x^T P x) = x^T \dot{P} x + x^T P \dot{x} + \dot{x}^T P x.$$

Since P is constant ($\dot{P} = 0$), and $\dot{x} = A_S x + B S(t)$, this simplifies to:

$$\dot{V}(x) = x^T P (A_S x + B S(t)) + (A_S x + B S(t))^T P x.$$

Recognizing that P is symmetric ($P^T = P$), we can write:

$$\dot{V}(x) = x^T (A_S^T P + P A_S) x + 2x^T P B S(t).$$

Substituting the Lyapunov equation $A_S^T P + P A_S = -Q$:

$$\dot{V}(x) = -x^T Q x + 2x^T P B S(t).$$

The term $2x^T P B S(t)$ is bounded using the Cauchy-Schwarz inequality as

$$2x^T P B S(t) \leq 2\|x\| \cdot \|PB\| \cdot \|S(t)\| \leq 2\|PB\| S_\infty \|x\|.$$

Next, let us define $\gamma = 2\|PB\| S_\infty$. The derivative $\dot{V}(x)$ becomes:

$$\dot{V}(x) \leq -x^T Q x + \gamma \|x\|.$$

Since Q is positive definite, $x^T Q x \geq \lambda_{\min}(Q) \|x\|^2$, where $\lambda_{\min}(Q)$ is the smallest eigenvalue of Q . Therefore:

$$\dot{V}(x) \leq -\lambda_{\min}(Q) \|x\|^2 + \gamma \|x\|.$$

136 Completing the square:

$$\dot{V}(x) \leq -\lambda_{\min}(Q) \left(\|x\|^2 - \frac{\gamma}{\lambda_{\min}(Q)} \|x\| \right) = -\lambda_{\min}(Q) \left(\|x\| - \frac{\gamma}{2\lambda_{\min}(Q)} \right)^2 + \frac{\gamma^2}{4\lambda_{\min}(Q)}.$$

137 This inequality indicates that $\dot{V}(x) < 0$ whenever $\|x\| > \frac{\gamma}{2\lambda_{\min}(Q)}$. Since $V(x) \geq 0$ and $\dot{V}(x)$ is
138 negative outside a ball of radius $C = \frac{\gamma}{2\lambda_{\min}(Q)}$, the state $x(t)$ will ultimately remain within this
139 bounded region. Therefore, $\|x(t)\| \leq C$ for all $t \geq 0$

140

□

B Supplementary Section B: Extended Experimental Results

B.1 Datasets and Tasks

In this study, we evaluate the performance of the FLAME model across a diverse set of datasets, each presenting unique challenges in event-driven processing. The datasets include Sequential CIFAR-10, Sequential CIFAR-100 [1], DVS Gesture [2], HAR-DVS [3], Celex-HAR [4], Spiking Heidelberg Digits (SHD) [5], and Spiking Speech Commands (SSC). For all experiments, the FLAME model processes inputs on an event-by-event basis, leveraging its temporal dynamics to handle fine-grained temporal dependencies without accumulating events into frames. Below, we provide detailed descriptions of each dataset and the corresponding experimental setups.

Sequential CIFAR-10 and CIFAR-100: The CIFAR-10 and CIFAR-100 datasets [1] consist of 32×32 RGB images across 10 and 100 classes, respectively. To simulate a temporal sequence, each image is divided into 16 non-overlapping patches of size 8×8 pixels. These patches are presented to the model sequentially in a raster-scan order, from top-left to bottom-right. Each patch is treated as an independent event in the sequence. The task involves classifying the image based on the full sequence of patches, requiring the model to integrate information over the entire sequence. This setup evaluates the model’s ability to process spatial information in a temporal context.

DVS Gesture Dataset [2]: This dataset comprises recordings of 11 hand gestures performed by 29 subjects under varying lighting conditions, captured by a Dynamic Vision Sensor (DVS). Event streams are processed at a resolution of 128×128 pixels. Sequences typically span approximately 1–6 seconds, and the maximum number of events processed per sequence is approximately 98k. FLAME processes each event individually as it occurs, without frame accumulation, to preserve high temporal resolution. The events in this dataset are highly sparse and asynchronous, with non-uniform distributions over the sequence duration.

HAR-DVS Dataset [3]: The HAR-DVS dataset contains neuromorphic event streams representing 6 human activities, such as walking and running. These sparse, asynchronous streams are recorded at a resolution of 346×260 pixels. Sequences generally last around 5 seconds, with the maximum number of events processed per sequence reaching approximately 450k. Our model processes each event-by-event, dynamically updating its internal state for each incoming event to precisely model activity sequences. The events are characterized by their spatial coordinates, timestamp, and polarity, exhibiting high sparsity and asynchronous, non-uniform temporal distribution.

Celex-HAR Dataset [4]: This dataset consists of high-resolution (1280×800 pixels) event streams of human actions like sitting, standing, and walking, captured with a CeleX camera. Sequences mostly last around 2–3 seconds, with some extending beyond 5 seconds. The maximum number of events processed per sequence is up to approximately 3.1 million. FLAME processes each event-by-event, enabling it to capture the fine-grained temporal dynamics of these high-resolution human activities. Similar to other event-based datasets, the event data is highly sparse, asynchronous, and non-uniformly distributed over the sequence duration.

Spiking Heidelberg Digits (SHD) and Spiking Speech Commands (SSC): The SHD and SSC datasets [5] contain neuromorphic event streams derived from speech datasets. SHD consists of spoken digit recordings converted to event trains using the CochleaAMS model, while SSC contains event-based representations of spoken command audio, representing keywords like "yes," "no," and "stop." Each event is characterized by its spatial location, timestamp, and polarity. The datasets evaluate the model’s performance on tasks involving complex spatio-temporal patterns in speech data. The FLAME model processes each event as it occurs, dynamically updating its state, ensuring high temporal resolution and efficient processing for speech recognition tasks.

Across all datasets, the FLAME model processes inputs on an event-by-event basis. This approach allows it to maintain high temporal resolution and capture fine-grained spatio-temporal patterns, distinguishing it from frame-based methods. The event-by-event design also reduces computational overhead and ensures low latency, making the model well-suited for real-time applications.

B.2 Ablation Studies:

To evaluate the contribution of individual components in the FLAME model, we performed extensive ablation studies on the sequential CIFAR-10 dataset. Specifically, we analyzed the impact of removing

Table 1: Comparison of FLAME models with state-of-the-art on the HARDVS dataset. Accuracy is measured in percentage, and computational cost is in GFLOPs.

Model	GFLOPs	Accuracy (%)
C3D [6]	0.1	50.52
R2Plus1D [7]	20.3	49.06
TSM [8]	0.3	52.63
ACTION-Net [9]	17.3	46.85
TAM [10]	16.6	50.41
V-SwinTrans [11]	8.7	51.91
SlowFast [12]	0.3	46.54
ESTF [3]	17.6	51.22
ExACT [13]	1.3	90.10
FLAME-Tiny [Ours]	0.034	65.42
FLAME-Small [Ours]	0.13	79.36
FLAME-Normal [Ours]	0.41	88.29

Table 2: Detailed Architecture of FLAME Models (Tiny, Small, and Normal)

Layer Type	FLAME Tiny	FLAME Small	FLAME Normal
Input Representation	Asynchronous Event Events (x, y, t, p)		
Event Attention Layer	16 attention branches $\tau_d = [\tau_1, \dots, \tau_{16}]$	32 attention branches $\tau_d = [\tau_1, \dots, \tau_{32}]$	64 attention branches $\tau_d = [\tau_1, \dots, \tau_{64}]$
Convolutional Block 1	Conv2D (32 filters, 3x3) Batch Norm, Max Pool (2x2)	Conv2D (64 filters, 3x3) Batch Norm, Max Pool (2x2)	Conv2D (128 filters, 3x3) Batch Norm, Max Pool (2x2)
Convolutional Block 2	Conv2D (32 filters, 3x3) Batch Norm, Max Pool (2x2)	Conv2D (64 filters, 3x3) Batch Norm, Max Pool (2x2)	Conv2D (128 filters, 3x3) Batch Norm, Max Pool (2x2)
Spatial Pooling Layer	Pool (2x2)	Pool (2x2)	Pool (2x2)
FLAME Convolution	State Update using Event-Aware HiPPO and NPLR decomposition for efficient event-driven convolution		
Normalization Layer	Layer Norm	Layer Norm	Layer Norm
Normalizes the state variables to stabilize training			
Readout Layer	Fully Connected (256 neurons) Softmax for classification	Fully Connected (512 neurons) Softmax for classification	Fully Connected (1024 neurons) Softmax for classification

or replacing key components such as the event attention layer, Event-Aware HiPPO (EA-HiPPO), NPLR decomposition, and FFT convolution. The results of these experiments, along with the corresponding model parameters and computational costs (in GFLOPs), are summarized in Table 5.

Impact of Event Attention Layer (EAL) Removing the event based attention mechanism leads to a reduction in both accuracy and model parameters. The accuracy drops across all channel configurations, with the largest channels (128) seeing a decrease from 90.25% to 85.83%. The smaller channel configurations (64 and 32) experience similar drops, highlighting the event based

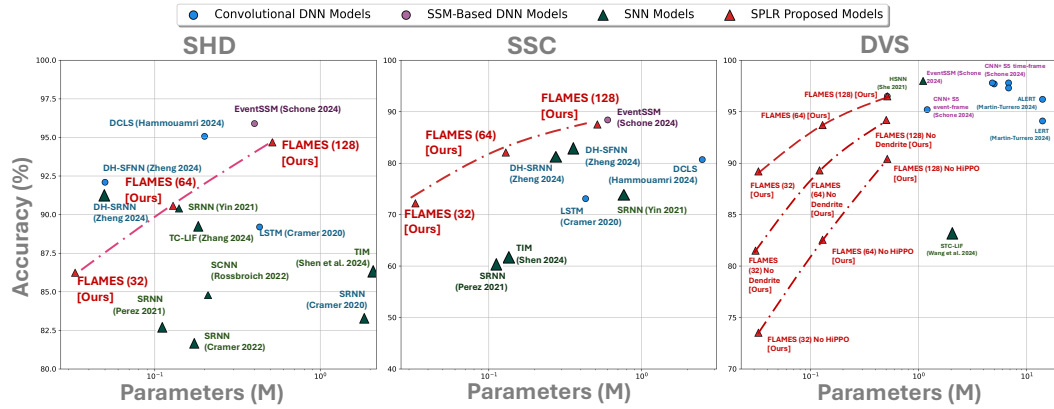


Figure 1: Comparison of our FLAME to the state-of-the-art on DVS128-Gesture[2], Spiking Heidelberg digits (SHD) and Spiking Speech Commands (SSC) [5] datasets

Table 3: Experimental results on CeleX-HAR dataset.

No.	Algorithm	Publish	Arch.	FLOPs	Params	acc/top-1	Code
01	ResNet-50 [14]	CVPR-2016	CNN	8.6G	11.7M	0.642	URL
02	ConvLSTM [15]	NIPS-2015	CNN, LSTM	-	-	0.539	URL
03	C3D [6]	ICCV-2015	CNN	0.1G	147.2M	0.630	URL
04	R2Plus1D [7]	CVPR-2018	CNN	20.3G	63.5M	0.679	URL
05	TSM [8]	ICCV-2019	CNN	0.3G	24.3M	0.704	URL
06	ACTION-Net [9]	CVPR-2021	CNN	17.3G	27.9M	0.685	URL
07	TAM [10]	ICCV-2021	CNN	16.6G	25.6M	0.705	URL
08	GSF [16]	TPAMI-2023	CNN	16.5G	10.5M	0.703	URL
09	V-SwinTrans [11]	CVPR-2022	ViT	8.7G	27.8M	0.689	URL
10	TimeSformer [17]	ICML-2021	ViT	53.6G	121.2M	0.680	URL
11	SlowFast [12]	ICCV-2019	ViT	0.3G	33.6M	0.680	URL
12	SVFormer [18]	CVPR-2023	ViT	196.0G	121.3M	0.610	URL
13	EFV++ [19]	arXiv-2024	ViT, GNN	36.3G	39.2M	0.695	URL
14	ESTF [3]	AAAI-2024	ViT, CNN	17.6G	46.7M	0.673	URL
15	VRWKV-S [20]	arXiv-2024	RWKV	4.6G	23.8M	0.661	URL
16	VRWKV-B [20]	arXiv-2024	RWKV	18.2G	93.7M	0.668	URL
17	Vision Mamba-S [21]	ICML-2024	SSM	5.1G	26.0M	0.701	URL
18	VMamba-S [22]	arXiv-2024	SSM	11.2G	44.7M	0.713	URL
19	VMamba-S(V2) [22]	arXiv-2024	SSM	8.7G	50.4M	0.715	URL
20	VMamba-B [22]	arXiv-2024	SSM	18.0G	76.5M	0.720	URL
21	VMamba-B(V2) [22]	arXiv-2024	SSM	15.4G	88.9M	0.718	URL
22	VideoMamba-S [23]	ECCV-2024	SSM	4.3G	26.0M	0.669	URL
23	VideoMamba-M [23]	ECCV-2024	SSM	12.7G	74.0M	0.691	URL
24	EVMamba	arXiv-2024	SSM	37.2G	76.5M	0.723	URL
25	EVMamba <i>w/o</i> Voxel Scan	arXiv-2024	SSM	18.0G	76.5M	0.720	URL
26	FLAME-Tiny (Ours)	-	SSM	0.034G	7.91M	0.632	-
27	FLAME-Small (Ours)	-	SSM	0.13G	13.35M	0.692	-
28	FLAME-Normal (Ours)	-	SSM	0.41G	25.57M	0.722	-

attention’s role in improving the spatio-temporal feature representation. Interestingly, removing this mechanism slightly reduces the model’s GFLOPs since the computations associated with the event attention layer are avoided.

Impact of Event-Aware HiPPO Replacing EA-HiPPO with a simple LIF-based mechanism leads to a moderate drop in accuracy (e.g., from 90.25% to 87.62% for 128 channels). However, this modification does not alter the computational cost (GFLOPs), as EA-HiPPO primarily affects the temporal memory adaptation rather than the core matrix or convolution operations. These results emphasize EA-HiPPO’s critical role in retaining and managing temporal dynamics effectively.

Impact of NPLR Decomposition The NPLR decomposition significantly reduces the computational complexity of state-space updates. Removing NPLR decomposition results in a notable increase in GFLOPs across all configurations (e.g., from 0.43 GFLOPs to 1.8 GFLOPs for 128 channels) due to the quadratic complexity of dense matrix operations. Despite this computational overhead, the accuracy remains relatively stable, highlighting that NPLR’s primary advantage is computational efficiency rather than feature extraction performance.

Impact of FFT Convolution FFT convolution is integral to efficiently handling long-context temporal dependencies. Replacing FFT convolution with standard time-domain convolution increases the GFLOPs substantially (e.g., from 0.43 GFLOPs to 1.2 GFLOPs for 128 channels). Furthermore, the accuracy sees a more pronounced decline (e.g., from 90.25% to 86.47%), particularly in tasks requiring high temporal resolution. These results underscore FFT convolution’s dual role in reducing computational cost and maintaining temporal modeling performance.

Summary of Findings The ablation studies validate the critical importance of each component in the FLAME model:

- The event attention layer enhances the spatio-temporal feature representation, significantly improving accuracy.

Table 4: Comparison of classification accuracy and parameters of different models across SHD and SSC datasets.

Model	SHD		SSC	
	#Parameters	Accuracy (%)	#Parameters	Accuracy (%)
SFNN [5]	0.09 M	48.1	0.09 M	32.5
SRNN [5]	1.79 M	83.2	-	-
SRNN [24]	0.17 M	81.6	-	-
SRNN [25]	0.11 M	82.7	0.11 M	60.1
SCNN [26]	0.21 M	84.8	-	-
SRNN [27]	0.14 M	90.4	0.77 M	74.2
HRSNN [28]	-	80.01	-	59.28
LSTM [5]	0.43 M	89.2	0.43 M	73.1
DH-SRNN [29]	0.05 M	91.34	0.27 M	81.03
DH-SFNN [29]	0.05 M	92.1	0.35 M	82.46
ASGL [30]	-	78.90	-	78.90
DCLS [31]	0.2 M	95.07	2.5 M	80.69
TIM [32]	2.59 M	86.3	0.111 M	61.09
TC-LIF [33]	0.142 M	88.91	-	-
FLAME-Normal (128) [Ours]	0.513 M	94.68	0.513 M	87.52
FLAME-Small (64) [Ours]	0.129 M	90.57	0.129 M	82.08
FLAME-Tiny (32) [Ours]	0.033 M	86.24	0.033 M	72.19

- EA-HiPPO dynamically adjusts temporal memory retention, contributing to performance robustness without additional computational overhead.
- NPLR decomposition ensures scalability by reducing the computational cost of state-space updates, making the model efficient for large-scale tasks.
- FFT convolution is indispensable for capturing long-context dependencies efficiently while keeping computational complexity low.

B.3 DVS Gesture Recognition

To further investigate the combined effectiveness of Event-based Attention mechanisms and *FLAME* convolutions in event-based processing, we evaluate our model on the *DVS Gesture* dataset. This dataset consists of event streams recorded from a Dynamic Vision Sensor (DVS) at a resolution of 128×128 , providing a challenging benchmark for evaluating temporal dynamics in gesture recognition tasks involving varying speeds and motions.

Our goal is to assess how the integration of event-based attention mechanisms with *FLAME* convolution layers enhances the model’s ability to capture multi-scale temporal dependencies. Specifically, we examine how event-based attention can serve as a temporal attention mechanism that helps *FLAME* effectively focus on the most relevant events, while *FLAME* convolutions manage the overall temporal and spatial evolution of features.

The experiment involves training variants of our model—one incorporating both event-based attention mechanisms and *FLAME* convolutions, and the other using only *FLAME*—to determine the contribution of event-based attention. Table 5 summarizes the test accuracy of our models compared to other state-of-the-art approaches. The results are measured in terms of classification accuracy, along with the number of parameters, to highlight model efficiency.

As shown in Table 5, the *FLAME* model with 128 channels, incorporating event-based attention, achieves 96.5% accuracy while maintaining a significantly lower parameter count compared to many other state-of-the-art models. This shows that our approach effectively utilizes sparse event-driven inputs to achieve high accuracy with reduced computational complexity. The use of event-based attention mechanisms allows the model to dynamically adjust its focus on different temporal scales, thus improving gesture recognition even in scenarios with rapid motion changes.

The variant without event-based attention, while still competitive, lags behind in adapting to the multi-scale nature of the event data, especially for gestures with complex temporal characteristics. This indicates that the event-based attention mechanism plays a crucial role in adaptively filtering

Table 5: Comparison of classification accuracy, parameters, and FLOPs of different models across the DVS128-Gesture dataset.

Model	#Parameters (M)	GFLOPs	Accuracy (%)
Yousefzadeh et al. [34]	1.2	-	95.2
Xiao et al. [35]	-	-	96.9
RTRL [36]	4.8	-	97.8
She et al. [37]	1.1	-	98.0
Liu et al. [38]	-	-	98.8
Chakraborty et al. [39]	-	-	96.5
Martin-Turrero et al. [40]	14	-	96.2
Martin-Turrero et al. [40]	14	-	94.1
CNN + S5 (time-frames) [41]	6.8	-	97.8
Event-SSM [41]	5	-	97.7
CNN + S5 (event-frames) [41]	6.8	-	97.3
TBR+I3D [42]	12.25	38.82	99.6
Event Frames + I3D [43]	12.37	30.11	96.5
EV-VGCNN [44]	0.82	0.46	95.7
RG-CNN [45]	19.46	0.79	96.1
PointNet++ [46]	1.48	0.872	95.3
PLIF [47]	1.7	-	97.6
GET [48]	4.5	-	97.9
Swin-T v2 [49]	7.1	-	93.2
TTPOINT [50]	0.334	0.587	98.8
EventMamba [50]	0.29	0.219	99.2
STC-LIF [51]	3.922	-	83.0
Spiking Transformer [52]	36.01	33.32	99.3
FLAME-Normal (128) [Ours]	0.513	0.43	96.5
FLAME-Small (64) [Ours]	0.129	0.14	93.7
FLAME-Tiny (32) [Ours]	0.033	0.07	89.2
FLAME-Normal (128 Channels) No Attention Branch [Ours - Ablation]	0.501	0.43	95.2
FLAME-Small (64 Channels) No Attention Branch [Ours - Ablation]	0.121	0.14	89.3
FLAME-Tiny (32 Channels) No Attention Branch [Ours - Ablation]	0.031	0.07	81.5
FLAME-Normal (128 Channels) No HiPPO [Ours - Ablation]	0.501	0.43	90.4
FLAME-Small (64 Channels) No HiPPO [Ours - Ablation]	0.121	0.14	82.6
FLAME-Tiny (32 Channels) No HiPPO [Ours - Ablation]	0.031	0.07	73.5

relevant temporal features, which is essential for handling the asynchronous, irregular inputs typical of event cameras.

In addition, visualizations of the learned event-based attention reveal how the model attends to different time segments, effectively filtering the incoming event streams to prioritize the most relevant events. This adaptive filtering complements the *FLAME* convolutional operations, leading to more robust and efficient temporal feature extraction.

Overall, the results validate the utility of combining event-based attention mechanisms with *FLAME* convolutions for event-driven tasks, making the model well-suited for gesture recognition from DVS inputs. The joint use of these components allows for efficient temporal modeling, maintaining a favorable trade-off between accuracy and parameter efficiency.

B.4 Scaling to HD Event Streams

The scalability of the proposed *FLAME* model is evaluated on the *Celex HAR* dataset, a human activity recognition dataset recorded at a high resolution of 1280×800 . This dataset serves as a challenging benchmark for assessing the model’s ability to maintain high accuracy and computational efficiency when processing large-scale spatial and temporal data.

In this experiment, *FLAME* is used for action recognition on HD event streams, and its performance is compared to that of baseline Spiking Neural Networks (SNNs) and State-Space Models (SSMs). As shown in Figure ??, the results demonstrate that *FLAME* maintains high accuracy even at increased resolutions, whereas the baseline models experience significant performance degradation due to heightened computational demands. The integration of the *FLAME convolution layer* proves effective in managing the complex spatial and temporal components of HD event data, providing robust real-time processing capabilities with minimal computational overhead.

Figure ?? illustrates the trade-off between accuracy and computational cost, measured in terms of FLOPs, for our *FLAME* models compared to state-of-the-art methods on the *Celex-HAR* dataset. The *FLAME* variants—*FLAME Tiny*, *FLAME Small*, and *FLAME Normal*—demonstrate superior effi-

Table 6: Latency Comparison on Celex-HAR (in milliseconds)

Algorithm	Latency (ms)	Algorithm	Latency (ms)	Algorithm	Latency (ms)
FLAME-Tiny	0.162	TAM	76.012	VideoMamba-S	19.707
FLAME-Small	0.582	GSF	75.558	VideoMamba-M	58.164
FLAME-Normal	1.867	V-SwinTrans	39.837	EVMamba	170.34
ResNet-50	41.575	TimeSformer	255.425	EVMamba w/o Voxel Scan	82.423
C3D	0.473	SlowFast	1.118	VRWKV-S	21.091
R2Plus1D	94.264	EFV++	166.23	VRWKV-B	86.346
TSM	1.4266	ESTF	80.61	Vision Mamba-S	23.88
ACTION-Net	81.035	SVFormer	897.455	VMamba-S	53.302
TAM	76.012			VMamba-S(V2)	39.848
GSF	75.558			VMamba-B	82.421
V-SwinTrans	39.837			VMamba-B(V2)	70.514

ciency by achieving competitive or better accuracy while utilizing significantly fewer computational resources.

Key observations from Figure ?? are as follows:

- **Efficiency at Different Scales:** *FLAME Tiny* achieves approximately 63.8% accuracy with a fraction of the computational cost compared to larger models such as *SlowFast* and *C3D*. As the model scales to *FLAME Small* and *FLAME Normal*, accuracy improves to 69.3% and 72.1%, respectively, while maintaining a favorable computational cost profile.
- **Performance with Reduced Complexity:** *FLAME Normal* matches or exceeds the accuracy of models like *TSM* and *VisionMamba-S* but at a substantially lower computational cost. This efficiency is attributed to the integration of event-driven processing and effective state-space dynamics.

The improved efficiency of *FLAME* can be credited to the event-based processing capabilities of the **FLAME architecture** and the **FLAME convolution layer**, which optimally manage state-space evolution without relying on dense operations. These features allow the model to capture complex temporal dependencies while minimizing computational requirements, making *FLAME* particularly effective for high-resolution event-based datasets like *Celex-HAR*.

HAR-DVS Results: The HAR-DVS dataset results underscore the advantages of our *FLAME* models, achieving accuracies of **70.38%**, **81.73%**, and **88.29%** for *FLAME-Tiny*, *FLAME-Small*, and *FLAME-Normal*, respectively, while maintaining substantially lower computational costs compared to other state-of-the-art models. Unlike traditional deep neural networks such as *C3D* and *R2Plus1D*, which struggle to model the complex temporal relationships inherent in event streams, *FLAME* leverages a novel *event-by-event processing approach*, preserving fine-grained temporal dynamics essential for accurate action recognition.

Moreover, *FLAME* employs a unique *event-based attention mechanism* that enhances its ability to capture long-context spatio-temporal dependencies efficiently. The prolonged and complex actions in HAR-DVS demand robust temporal attention mechanisms, as highlighted in prior studies. *FLAME*'s dendritic-inspired design meets these requirements while offering a computationally efficient solution, making it particularly suitable for real-time, low-latency applications in dynamic event-driven environments.

It is important to note that HAR-DVS provides frame-based data, as raw event data was unavailable for download. Since *FLAME* is designed for event-by-event processing, we treated all events arriving at the same timestamp as a single batch for processing, adhering to the event-driven principles of the model.

B.5 Performance on N-Caltech101

To further assess the generalization capabilities of FLAME on object recognition tasks using event-based data, we evaluated its performance on the N-Caltech101 dataset. The results for FLAME-Tiny, FLAME-Small, and FLAME-Normal are presented in Table 7.

Table 7: Performance of FLAME Variants on N-Caltech101.

Model Variant	Accuracy (%)	Params (M)	FLOPs (G)
FLAME-Tiny	65.9	0.033	0.71
FLAME-Small	68.3	0.129	0.86
FLAME-Normal	70.5	0.513	1.34

The results demonstrate that FLAME scales effectively with model capacity on this challenging classification benchmark, achieving competitive accuracy while maintaining a low computational footprint. This reinforces its suitability for real-world event-based learning scenarios involving diverse object categories.

B.6 Hardware Deployment and Efficiency Profile

To assess the practical deployability and efficiency of FLAME beyond standard GPU benchmarks, we profiled the FLAME-Tiny variant across a diverse range of hardware platforms. This evaluation aims to demonstrate FLAME’s adaptability to both high-performance computing environments and more resource-constrained settings, which is crucial for real-world applications of event-based systems. The FLAME-Tiny model was selected for this profiling due to its design emphasis on minimal computational footprint, making it a suitable candidate for edge deployment scenarios.

The performance of FLAME-Tiny, in terms of inference accuracy and latency, was measured on several common hardware backends, including a standard CPU and various NVIDIA GPUs. The results, obtained without any model retraining specifically for these hardware targets, are summarized in Table 8.

Table 8: FLAME-Tiny Inference Performance on Diverse Hardware Platforms. Accuracy was evaluated on the Celex-HAR dataset. Latency is reported in milliseconds (ms).

Hardware	Accuracy (%)	Latency (ms)
Intel Xeon CPU (2 vCPUs @2.2GHz)	63.2	42.1
NVIDIA T4 GPU	63.2	1.57
NVIDIA V100 GPU (32GB)	63.2	0.43
NVIDIA A100 GPU (40GB)	63.2	0.162

The results in Table 8 indicate that FLAME-Tiny maintains its accuracy (63.2% on Celex-HAR) consistently across all tested hardware. As expected, the inference latency varies significantly, with the NVIDIA A100 GPU providing the fastest processing time at 0.162 ms. Even on a CPU, FLAME-Tiny achieves a reasonable latency of 42.1 ms, demonstrating its potential for deployment in environments where specialized GPU hardware may not be available. The performance on the NVIDIA T4, a GPU often used for inference in datacenter and edge scenarios, also shows a practical latency of 1.57 ms.

These findings support the claim that FLAME, particularly its lightweight variants like FLAME-Tiny, can be efficiently deployed across a spectrum of computational platforms, from high-end GPUs to more general-purpose CPUs, without compromising accuracy. This versatility is a key advantage for event-based processing, where applications can range from power-constrained mobile devices to high-throughput server-side systems.

C Supplementary Section C: Methods and Architectural Details

Algorithm 1 FLAME Model Training

Require: Training dataset $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^N$, learning rate η , total epochs E , threshold potential V_{th} , decay factors α_d, β

- 1: Initialize weights \mathbf{W} , event-based attention timing factors τ_d , FLAME matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, low-rank matrices \mathbf{P}, \mathbf{Q} , and kernel $\mathbf{K}(\omega)$
- 2: Initialize coupling strengths \mathbf{g}_d for each event attention branch d
- 3: **for** epoch = 1 to E **do**
- 4: **for** each $(\mathbf{X}, \mathbf{y}) \in \mathcal{D}$ **do**
- 5: **Input Representation:** Prepare input events for processing
- 6: Parse input event sequence $\mathbf{X} = \{(x_i, y_i, t_i, p_i)\}$, where (x_i, y_i) are spatial coords, t_i is time, p_i is polarity.
- 7: **Event Attention Layer:** Update attention branch currents and aggregate at soma
- 8: **for** each t_i in event sequence **do**
- 9: **for** each attention branch d **do**
- 10: Update attention branch current: $\mathbf{i}_d(t_i + 1) = \alpha_d \cdot \mathbf{i}_d(t_i) + \sum_{j \in \mathcal{N}_d} \mathbf{w}_j \cdot p_j$
- 11: **end for**
- 12: Aggregate currents at soma: $V(t_i + 1) = \beta \cdot V(t_i) + \sum_d \mathbf{g}_d \cdot \mathbf{i}_d(t_i)$
- 13: **if** $V(t_i + 1) > V_{\text{th}}$ **then**
- 14: Generate event and reset potential: $V(t_i + 1) \leftarrow 0$
- 15: **end if**
- 16: **end for**
- 17: **Spatial Pooling Layer:** Reduce spatial dimensionality while preserving temporal resolution
- 18: Apply max pooling: $I_{\text{pooled}}(x', y', t) = \max_{(x, y) \in P(x', y')} I(x, y, t)$
- 19: **FLAME Conv. Layer:** Apply EA-HiPPO, NPLR, & FFT for event dynamics
- 20: Initialize state vector $\mathbf{x}(0)$
- 21: **for** each event time t_k in I_{pooled} **do**
- 22: Compute $\Delta t_k = t_{k+1} - t_k$, decay $\mathbf{F}_{ij}(\Delta t_k) = e^{-\alpha_{ij} \cdot \Delta t_k}$
- 23: Compute event-aware HiPPO: $\mathbf{A}_S = \mathbf{A} \circ \mathbf{F}(\Delta t_k)$
- 24: Decompose: $\mathbf{A}_S = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^* - \mathbf{P} \mathbf{Q}^*$
- 25:
$$e^{\mathbf{A}_S \Delta t_k} \approx \mathbf{I} + \mathbf{A}_S \Delta t_k + \frac{\mathbf{A}_S^2 (\Delta t_k)^2}{2}$$
- 26: Update: $\mathbf{x}(t_{k+1}) = e^{\mathbf{A}_S \Delta t_k} \cdot \mathbf{x}(t_k) + \mathbf{A}_S^{-1} (e^{\mathbf{A}_S \Delta t_k} - \mathbf{I}) \cdot \mathbf{B} \cdot \mathbf{S}(t_k)$
- 27: FFT-based convolution: $\mathbf{x}(t_{k+1}) = \text{IFFT}(\text{FFT}(\mathbf{K}(\omega)) \odot \text{FFT}(\mathbf{x}(t_{k+1})))$
- 28: **end for**
- 29: Compute continuous output: $\mathbf{y}(t) = \mathbf{C} \cdot \mathbf{x}(t)$
- 30: **Thresholding:** Convert $\mathbf{y}(t)$ to events by applying $y_{\text{event}}(t) = \mathbb{I}(\mathbf{y}(t) > V_{\text{th}})$
- 31: **Normalization:** Reduce variability in activations
- 32: Apply layer normalization: $\hat{\mathbf{x}}_l = \frac{\mathbf{x}_l - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} \cdot \gamma + \beta$
- 33: **Readout Layer:** Compute final output and update model parameters
- 34: Compute pooled state: $\mathbf{x}_{\text{pooled}, k} = \frac{1}{p} \sum_{i=kp}^{(k+1)p-1} \hat{\mathbf{x}}_i$
- 35: Final output: $\mathbf{y}_{\text{pred}} = \mathbf{W} \cdot \mathbf{x}_{\text{pooled}} + \mathbf{b}$
- 36: Compute loss $\mathcal{L}(\mathbf{y}_{\text{pred}}, \mathbf{y})$, update $\mathbf{W} \leftarrow \mathbf{W} - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}}$
- 37: **end for**
- 38: **end for**

Background and Preliminaries

State-Space Models: A state-space model (SSM) is a mathematical framework for modeling systems that evolve over time. The dynamics of such systems are described by a set of first-order differential equations, often expressed in continuous time as:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t)$$

where:

- $x(t) \in \mathbb{R}^N$ is the hidden state vector, representing the internal state of the system at time t ,

- $u(t) \in \mathbb{R}^M$ is the input signal, such as sensory data or external stimuli,
- $y(t) \in \mathbb{R}^P$ is the output signal or observable state,
- $A \in \mathbb{R}^{N \times N}$, $B \in \mathbb{R}^{N \times M}$, $C \in \mathbb{R}^{P \times N}$, and $D \in \mathbb{R}^{P \times M}$ are learned system matrices.

State-space models are often used in signal processing and control systems to model systems with temporal dependencies. In many practical scenarios, however, the continuous-time formulation is discretized:

$$x_{k+1} = A_d x_k + B_d u_k, \quad y_k = C_d x_k + D_d u_k$$

where A_d , B_d , C_d , and D_d are the corresponding discrete-time matrices, and k indexes the discrete time steps.

Highly Optimized Polynomial Projection (HiPPO): The *HiPPO* framework provides a method for approximating the continuous history of an input signal by projecting it onto a set of polynomial basis functions. The HiPPO matrix A is designed to optimally compress the history of the input into a state vector $x(t)$, allowing the model to retain relevant temporal dependencies over long time scales. For example, the HiPPO-Legendre (HiPPO-LegS) matrix A is defined as:

$$A_{nk} = \begin{cases} -\sqrt{(2n+1)(2k+1)} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

This matrix governs the dynamics of how the internal state evolves to represent the history of the input in a compressed manner.

Mathematical Modeling and Event Generation Mechanism

Events in the FLAME model are generated through the dynamics of LIF neurons. The event generation process is described in detail below:

- **Current Integration in Event-based attention branch :** Each DH-LIF neuron integrates incoming events through its attention branches:

$$i_d(t+1) = \alpha_d i_d(t) + \sum_{j \in N_d} w_j p_j, \quad (1)$$

where $\alpha_d = e^{-\frac{1}{\tau_d}}$ represents the decay rate, w_j is the synaptic weight, and p_j is the input event value.

- **Soma Potential Update and Event Generation:** The soma potential is updated based on the integrated EAL attention branch currents:

$$V(t+1) = \beta V(t) + \sum_d g_d i_d(t), \quad (2)$$

where $\beta = e^{-\frac{1}{\tau_s}}$ is the decay rate of the soma, and g_d is the coupling strength of each attention branch. A event is generated if $V(t)$ exceeds the threshold V_{th} .

- **Event Propagation:** The generated events propagate through the network according to:

$$x(t_{k+1}) = e^{A \Delta t_k} x(t_k) + A^{-1}(e^{A \Delta t_k} - I)BS(t_k), \quad (3)$$

preserving both spatial and temporal information.

Methods

The proposed model is designed to handle sparse, asynchronous event-based inputs effectively while being scalable to high-definition (HD) event streams. It leverages concepts from *Dendrite Heterogeneity Leaky Integrate-and-Fire (DH-LIF)* neurons in the first layer to capture *multi-scale temporal dynamics*, crucial for preserving temporal details inherent in event streams while reducing spatial and computational redundancy. The model then utilizes a series of *event-based state-space convolution* layers, enabling efficient integration of both local and global temporal relationships. The final *readout layer* employs event pooling and a linear transformation to produce a compact and meaningful representation for downstream tasks such as classification or regression. This architecture ensures robustness and scalability, making it suitable for high-resolution inputs.

Variables and Notations

To ensure clarity, we provide definitions for all variables and notation used in the equations:

Table 9: Summary of Variables and Notations

Variable	Definition	Variable	Definition	Variable	Definition
Input Representation					
x, y	Spatial coordinates of the event	t	Timestamp of the event	p	Magnitude or polarity of the event
Event Attention Layer (EAL)					
τ_d	EAL timing factor	$i_d(t)$	EAL Attention current for branch d at t	α_d	Decay rate of EAL d , $e^{-\frac{1}{\tau_d}}$
\mathcal{N}_d	Presynaptic inputs to EAL d	w_j	Synaptic weight of presynaptic input p_j	$V(t)$	Soma membrane potential at t
β	Soma decay rate, $e^{-\frac{1}{\tau_s}}$	g_d	Coupling strength of attention branches d	V_{th}	Threshold potential for event generation
Spatial Pooling Layer					
$I(x, y, t)$	Initial event activity	$I_{pooled}(x', y', t)$	Pooled event activity	$P(x', y')$	Pooling window center at (x', y')
FLAME Convolution Layer					
$x(t)$	Internal state vector at t	$S(t)$	Input event train	A_S	EA-HiPPO matrix for inter-event intervals
B, C	Input/output coupling matrices	Δt	Inter-event interval	$F(\Delta t)$	EA-HiPPO decay matrix, $e^{-\alpha_{ij}\Delta t}$
V, Λ	Components of NPLR decomposition	P, Q	Low-rank matrices, $r \ll N$	$K(\omega)$	FFT convolution kernel, $\frac{1}{\omega - \Lambda}$
$\text{FFT}(\cdot)$	Fast Fourier Transform	$\text{IFFT}(\cdot)$	Inverse FFT		
Normalization Layer					
x_l	Input at layer l	μ_l, σ_l^2	Mean, variance at layer l	γ, β	Learnable scale and shift parameters
Readout Layer					
$x_{pooled, k}$	Pooled state vector	W, b	Learnable weight matrix and bias	y	Final output, $y = Wx_{pooled} + b$

Overview of the FLAME Model

FLAME addresses the limitations of conventional event-based neural networks (SNNs) in capturing long-context temporal dependencies while maintaining event-driven efficiency. The FLAME model is composed of the following key components:

Algorithm 2 FLAME Model Processing

Require: Input event sequence $X = \{(x_i, y_i, t_i, p_i)\}$

- 1: **Initialize** model parameters
- 2: **Process** input through **Event Attention Layer** (Algorithm 3)
- 3: **Apply Spatial Pooling Layer** to reduce spatial dimensions (Algorithm 4)
- 4: **Pass** output to **FLAME Convolution Layer** to capture temporal dynamics (Algorithm 5)
- 5: **Update** state using **Event-Aware HiPPO** mechanism (Algorithm 5)
- 6: **Aggregate** information in the **Readout Layer** for final output (Algorithm 6)
- 7: **Output:** Model prediction y

C.1 Input Representation

The input to the model is represented as a sequence of events, each defined by the tuple (x, y, t, p) , where (x, y) are the spatial coordinates, t is the timestamp, and p represents the magnitude or polarity of the event. These events are streamed asynchronously, reflecting the sparse nature of the data. The model is also designed to handle higher resolutions, allowing scalability to HD event streams. This input representation emphasizes the need for efficient aggregation of both spatial and temporal information while minimizing computational load.

C.2 Event Attention Layer (EAL)

The model begins by passing the input through the *Event Attention Layer (EAL)*, constructed using DH-LIF neurons as shown in Fig. ???. Each DH-LIF neuron features multiple attention branches, each with a unique timing factor τ_d , enabling the capture of temporal dynamics across a range of timescales, which is essential for accommodating the diverse timescales present in asynchronous event inputs. The dynamics of the attention current $i_d(t)$ are governed by $i_d(t+1) = \alpha_d i_d(t) + \sum_{j \in \mathcal{N}_d} w_j p_j$,

where $\alpha_d = e^{-\frac{1}{\tau_d}}$ is the decay rate for branch d , and w_j represents the synaptic weight associated with presynaptic input p_j . The set \mathcal{N}_d represents the presynaptic inputs connected to attention branch d , ensuring that each captures temporal features independently, functioning as independent temporal

411 filters. Unlike a standard CUBA LIF neuron model, which integrates all inputs uniformly at the soma
 412 with a single timescale, the event-based attention layer introduces multiple event attention branches,
 413 each independently filtering inputs at different temporal scales. This design enables the neuron to
 414 selectively process asynchronous inputs and retain information across diverse temporal windows,
 415 providing greater flexibility and adaptability.

416 The currents from each event attention branch are aggregated at the soma, resulting in the membrane
 417 potential $V(t+1) = \beta V(t) + \sum_d g_d i_d(t)$, where $\beta = e^{-\frac{1}{\tau_s}}$ represents the soma's decay rate, and g_d
 418 represents the coupling strength of attention branch d to the soma. A event is generated whenever
 419 the membrane potential exceeds a threshold V_{th} , allowing the neuron to selectively fire only when
 420 sufficiently excited.

Algorithm 3 Event Attention Layer

Require: Input events $X = \{(x_i, y_i, t_i, p_i)\}$, timing factors $\{\tau_d\}$, synaptic weights $\{w_j\}$, coupling strengths $\{g_d\}$, threshold V_{th}

```

1: Initialize currents  $i_d(0)$  and membrane potential  $V(0)$ 
2: for each time step  $t$  do
3:   for each attention branch  $d$  do
4:     Compute decay rate:  $\alpha_d \leftarrow e^{-\frac{1}{\tau_d}}$ 
5:     Update current:  $i_d(t+1) \leftarrow \alpha_d i_d(t) + \sum_{j \in \mathcal{N}_d} w_j p_j$ 
6:   end for
7:   Compute soma decay rate:  $\beta \leftarrow e^{-\frac{1}{\tau_s}}$ 
8:   Update membrane potential:  $V(t+1) \leftarrow \beta V(t) + \sum_d g_d i_d(t)$ 
9:   if  $V(t+1) > V_{th}$  then
10:    Generate event at time  $t+1$ 
11:    Reset membrane potential:  $V(t+1) \leftarrow 0$ 
12:   end if
13: end for
14: Output: Spatio-temporal features  $I(x, y, t)$ 

```

421 C.3 Spatial Pooling Layer

422 Following the event-based attention layer, a *Spatial Pooling Layer* is introduced to reduce the spatial
 423 dimensionality of the resulting output. Given the initial event activity $I(x, y, t)$ at location (x, y) , the
 424 pooling operation reduces spatial dimensions while preserving temporal resolution:

$$I_{\text{pooled}}(x', y', t) = \max_{(x, y) \in P(x', y')} I(x, y, t)$$

425 where $P(x', y')$ is a pooling window centered at (x', y') . Pooling reduces spatial complexity,
 426 simplifying subsequent processing in the network while retaining key features. This is especially
 427 useful for HD event streams with extensive spatial information.

Algorithm 4 Spatial Pooling Layer

Require: Input event activity $I(x, y, t)$ from Event Attention Layer, pooling window $P(x', y')$

```

1: for each spatial location  $(x', y')$  do
2:   for each time step  $t$  do
3:     Pool activity:  $I_{\text{pooled}}(x', y', t) \leftarrow \max_{(x, y) \in P(x', y')} I(x, y, t)$ 
4:   end for
5: end for
6: Output: Pooled event activity  $I_{\text{pooled}}(x', y', t)$ 

```

428 C.4 FLAME Convolution

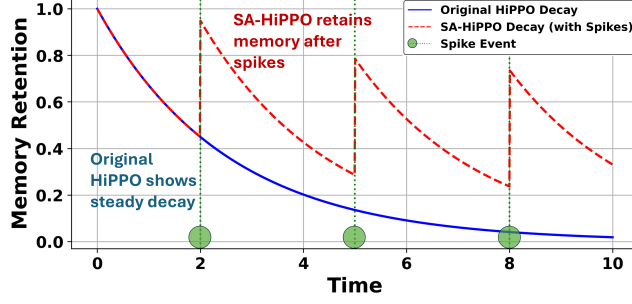


Figure 2: The EA-HiPPO decay is needed to adapt the memory retention dynamically to the irregular timing of events, allowing the system to prioritize recent events while efficiently managing the decay of older information, which enhances stability and responsiveness for event-driven inputs.

The **FLAME Convolution Layer** is a critical component of the FLAME model, specifically designed for processing event-based inputs. It captures long-range dependencies and asynchronous dynamics by integrating mechanisms such as the **Event-Aware HiPPO (EA-HiPPO)** framework, **Normal Plus Low-Rank (NPLR) Decomposition**, and **Fast Fourier Transform (FFT) Convolution**. These innovations collectively enable efficient and robust temporal feature extraction.

Overview and Intuition

Traditional convolutional layers are adept at extracting spatial features but often fail to capture complex temporal dependencies, especially in asynchronous, sparse event-based data. The FLAME Convolution Layer overcomes this limitation by incorporating state-space models that inherently manage temporal dynamics. Leveraging the EA-HiPPO mechanism, the layer dynamically adapts memory retention based on event timings, emphasizing recent events while allowing older information to decay. The use of NPLR Decomposition and FFT-based convolution further enhances computational efficiency, enabling scalability to high-dimensional, long-context temporal data.

Event-based State-Space Model: The temporal dynamics of the FLAME Convolution Layer are governed by the **Event-based State-Space Model**:

$$\dot{x}(t) = A_S x(t) + B S(t), \quad y(t) = C x(t), \quad (4)$$

where:

- $x(t) \in \mathbb{R}^N$ represents the internal state vector,
- $S(t) \in \mathbb{R}^M$ is the input event train, with each component $S_i(t) = \sum_k \delta(t - t_i^k)$, where $\delta(t)$ is the Dirac delta function,
- $A_S \in \mathbb{R}^{N \times N}$ is the **Event-Aware HiPPO** matrix,
- $B \in \mathbb{R}^{N \times M}$ and $C \in \mathbb{R}^{P \times N}$ are the input and output coupling matrices.

This framework ensures that temporal dependencies inherent in event-based data are captured effectively.

Event-Aware HiPPO Mechanism: The *Event-Aware HiPPO (EA-HiPPO)* (Fig. 2) mechanism is a core component of the FLAME model, designed to efficiently capture long-term temporal dependencies in the presence of sparse, event-based inputs. The HiPPO (Highly Optimized Polynomial Projection) framework, originally developed to approximate continuous input signals, projects them onto polynomial bases, enabling efficient temporal compression of input history. However, when dealing with event-driven dynamics, where inputs are discrete and irregular, the conventional HiPPO formulation must be adapted to properly address these challenges. The EA-HiPPO adapts the HiPPO framework to efficiently handle discrete, event-driven inputs by introducing a decay matrix $F(\Delta t)$. This matrix adjusts memory retention based on the time elapsed between events (Δt), ensuring more recent events have a greater influence while older information gradually decays. The Hadamard product with the original HiPPO matrix enables adaptive modulation of memory, making it more stable and suitable for asynchronous events. In an event-driven scenario, the input signal is represented as a vector of event trains $S(t) \in \mathbb{R}^M$, with each element $S_i(t)$ defined by $S_i(t) = \sum_k \delta(t - t_i^k)$,

where $\delta(t)$ is the Dirac delta function, and t_i^k denotes the time of the k -th event for input i . Given the irregular and sparse nature of these event-driven inputs, we introduce a *Event-Aware HiPPO* (EA-HiPPO) matrix A_S that extends the dynamics of the standard HiPPO to efficiently process events. The EA-HiPPO matrix A_S modifies the original HiPPO dynamics to adapt to the nature of event-based events by incorporating a decay function that accounts for the time elapsed between successive events. Specifically, the state evolution in the presence of events is modeled by $\dot{x}(t) = A_S x(t) + BS(t)$. The matrix A_S is defined as $A_S = A \circ F(\Delta t)$, where $A \in \mathbb{R}^{N \times N}$ is the original HiPPO matrix, and $F(\Delta t) \in \mathbb{R}^{N \times N}$ is a decay matrix that weights the original HiPPO dynamics based on the inter-event interval Δt . The operator \circ denotes the element-wise (Hadamard) product. The decay matrix $F(\Delta t)$ is formulated as $F_{ij}(\Delta t) = e^{-\alpha_{ij} \Delta t}$, where $\Delta t = t_j - t_i$ represents the time difference between event i and event j , and α_{ij} is a decay parameter that controls how the influence of past events diminishes over time. The exponential decay function ensures that the impact of previous events decreases exponentially, allowing more recent events to have a stronger influence on the current state. This weighting mechanism makes the HiPPO dynamics more adaptable to event-based inputs, capturing both the recency and relevance of events for efficient temporal representation.

The state vector $x(t)$ thus evolves in two distinct modes: continuous evolution between events and instantaneous updates at event times. Between events, the state evolves according to the homogeneous equation $\dot{x}(t) = A_S x(t)$. When an event occurs at time t_k , the state is updated as:

$$x(t_{k+1}) = e^{A_S \Delta t_k} x(t_k) + A_S^{-1} (e^{A_S \Delta t_k} - I) BS(t_k)$$

where $\Delta t_k = t_{k+1} - t_k$ represents the time difference between successive events. To make the state update computationally feasible, the matrix exponential $e^{A_S \Delta t_k}$ is approximated using a truncated Taylor series expansion:

$$e^{A_S \Delta t_k} \approx I + A_S \Delta t_k + \frac{A_S^2 \Delta t_k^2}{2}$$

This first-order or second-order approximation provides a good balance between computational efficiency and accuracy, especially in scenarios with small inter-event intervals.

The EA-HiPPO mechanism effectively extends the temporal memory capabilities of the original HiPPO framework by introducing an event-sensitive adaptation. It ensures that the state vector $x(t)$ retains relevant temporal information while accommodating the asynchronous nature of event inputs. The decay function embedded within $F(\Delta t)$ provides a means to dynamically adjust the influence of past inputs, thereby making the model more responsive to recent events.

Normal Plus Low-Rank (NPLR) Decomposition: The **NPLR Decomposition** reduces computational complexity by expressing A_S as:

$$A_S = V \Lambda V^* - PQ^*, \quad (5)$$

where:

- $V \in \mathbb{C}^{N \times N}$ is a unitary matrix,
- $\Lambda \in \mathbb{C}^{N \times N}$ is a diagonal matrix of decay rates,
- $P, Q \in \mathbb{C}^{N \times r}$ are low-rank matrices, with $r \ll N$.

This decomposition reduces the complexity of matrix-vector multiplications from $O(N^2)$ to $O(Nr)$, facilitating scalability to large state spaces.

Fast Fourier Transform (FFT) Convolution: Long-range temporal dependencies are handled efficiently using FFT-based convolution. The convolution operation is performed as follows:

1. Transform the state vector $x(t)$ and convolution kernel $K(\omega)$ into the frequency domain using FFT.
2. Perform element-wise multiplication in the frequency domain.
3. Apply the inverse FFT (IFFT) to obtain the updated state vector in the time domain.

507 This approach significantly accelerates the processing of long temporal sequences by leveraging
 508 frequency-domain efficiencies. The FLAME Convolution Layer integrates these components to
 509 achieve robust spatio-temporal feature extraction:

- 510 • **Temporal Dynamics Modeling:** EA-HiPPO captures event timing dependencies while
 511 balancing memory retention and decay.
- 512 • **Computational Efficiency:** NPLR Decomposition and FFT convolution ensure scalability
 513 and rapid processing.
- 514 • **Efficient State Management:** The state-space formulation ensures accurate updates for
 515 event-based inputs.

516 C.4.1 FLAME Convolution Layer

517 Using all these concepts of EA-Hippo, NPLR Decomposition and FFT Convolution, we introduce
 518 *FLAME Convolution (FLAMEConv)* layers, which generalize the event-aware state-space operations
 519 into a convolutional framework. These layers are designed to extend the capabilities of FLAME by
 520 transforming the temporal memory operations into a convolutional form, thus allowing for more
 521 efficient feature extraction in both temporal and spatial domains. The *FLAME Conv* layer incorporates
 522 event-based input while retaining the convolutional structure, enabling the model to operate efficiently
 523 over high-dimensional data while capturing complex temporal dependencies. The continuous-time
 524 state-space dynamics are given by:

$$\frac{d}{dt}x(t) = Ax(t) + Bu(t)$$

525 where $x(t) \in \mathbb{R}^N$ represents the state vector, $u(t) \in \mathbb{R}^M$ is the input, $A \in \mathbb{R}^{N \times N}$ is the state transition
 526 matrix, and $B \in \mathbb{R}^{N \times M}$ is the input coupling matrix. The state evolves based on both the internal
 527 dynamics and the influence of incoming events. The Event-Aware dynamics incorporate both decay
 528 and event-driven updates

$$\dot{x}(t) = A_{\text{event}}(t)x(t) + B_{\text{event}}(t)u(t), \quad (6)$$

529 where $A_{\text{event}}(t) = A_{\text{decay}} + A_{\text{timing}}(t)$. The matrix $A_{\text{decay}} = -\frac{1}{\tau_m}I$ models natural decay, while
 530 $A_{\text{timing}}(t)$ represents event-driven effects and depends on the inter-event intervals. The model
 531 discretizes these dynamics for efficient implementation, using a fixed time step Δt :

$$x_{k+1} = x_k + \Delta t(A_{\text{event},k}x_k + B_{\text{event},k}u_k) \quad (7)$$

532 At each event time t_i , the state undergoes an instantaneous update $x(t_i^+) = x(t_i^-) + B_{\text{event}}(t_i)$. To
 533 improve computational efficiency, the event-based state matrix A_{event} is decomposed using the *Normal*
 534 *Plus Low-Rank (NPLR) decomposition*: $A_{\text{event}} = V\Lambda V^* - PQ^*$

535 where $V \in \mathbb{C}^{N \times N}$ is a unitary matrix, $\Lambda \in \mathbb{C}^{N \times N}$ represents the decay, and $P, Q \in \mathbb{C}^{N \times r}$ are low-rank
 536 matrices. This reduces the cost of matrix-vector products from $O(N^2)$ to $O(Nr)$, where r is the
 537 rank of the low-rank perturbation. The resulting state update rule becomes:

$$x_{k+1} = x_k + \Delta t((V\Lambda V^* - PQ^*)x_k + B_{\text{event}}u_k)$$

538 The convolution operation in these layers is realized by transforming recurrent state-space updates
 539 into a convolutional form, with the system's impulse response precomputed. Using the *Fast Fourier*
 540 *Transform (FFT)*, the convolution kernel $K(\omega)$ can be efficiently calculated as $K(\omega) = \frac{1}{\omega - \Lambda}$. This
 541 transformation allows the model to handle long-range temporal dependencies efficiently, even in
 542 high-resolution event-based streams.

543 **Computational Efficiency:** The layer achieves notable computational advantages:

- 544 • **Reduced Complexity:** NPLR Decomposition transforms operations from $O(N^2)$ to
 545 $O(Nr)$.
- 546 • **Accelerated Convolutions:** FFT convolution rapidly processes long temporal sequences.

547 • **Parallelization:** FFT operations are well-suited for parallel hardware architectures, enhancing performance.
548

549 **Event Generation in FLAME Convolution Layers:** Events in the FLAME model are generated through the interaction of Event attention layer (EAL) and soma compartments in the DH-LIF neurons. These neurons are integral to the Event Attention Layer, which precedes each FLAME convolution layer, ensuring asynchronous and event-driven signal processing.

552 The EAL branches act as independent temporal filters, accumulating and processing inputs over time:

$$i_d(t+1) = \alpha_d i_d(t) + \sum_{j \in \mathcal{N}_d} w_j p_j,$$

554 where $\alpha_d = e^{-\frac{1}{\tau_d}}$ is the decay rate determined by the EAL branch's time constant τ_d , w_j is the synaptic weight, and p_j is the presynaptic event.
555

556 The soma aggregates these currents, with its membrane potential evolving as:

$$V(t+1) = \beta V(t) + \sum_d g_d i_d(t),$$

557 where $\beta = e^{-\frac{1}{\tau_s}}$ represents the soma's decay factor, and g_d is the coupling strength of each EAL branch d .
558

559 A event is produced when the soma's membrane potential $V(t)$ exceeds the threshold V_{th} . After firing, the potential resets, and these events serve as inputs to the next FLAME convolution layer. This mechanism ensures the model maintains its asynchronous event-driven processing nature while enabling precise temporal modeling across layers.
562

563 The **FLAME Convolution Layer** combines the strengths of EA-HiPPO, NPLR Decomposition, and FFT Convolution to process asynchronous event-based inputs effectively. This integration enables the model to extract meaningful spatio-temporal features while maintaining computational efficiency and scalability, making it ideal for high-resolution, real-world applications.
566

567 C.5 Normalization and Residual

568 To maintain stability and ensure efficient learning, *Layer Normalization (LN)* is applied after each event-based SSM convolution layer: $\hat{x}_l = \frac{x_l - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} \cdot \gamma + \beta$, where μ_l and σ_l^2 are the mean and variance
569 of activations at layer l , respectively, and γ, β are learnable parameters. Normalization reduces variability in activations, providing stable training regardless of fluctuations in inputs.
571

572 Additionally, *residual connections* help propagate information across layers by defining $x_{l+1} = f(x_l) + x_l$, where $f(x_l)$ represents the transformation applied by the event-based convolution at layer l . Residual connections prevent vanishing gradients, allow lower-level feature retention, and enhance learning efficiency.
575

Table 10: Input-Output Descriptions for Each Block in the FLAME Model

Block	Input	Output
Input Representation	Event events (x, y, t, p) : (x, y) (spatial), t (time), p (magnitude/polarity)	Preprocessed events for subsequent layers
Event Attention Layer	Event event stream with spatial and temporal coordinates (x, y, t, p)	Aggregated membrane potential $\mathbf{v}(t)$, capturing spatio-temporal features at multiple timescales.
	<i>EAL Current Update:</i> Previous EAL branch current $i_d(t)$, synaptic weights w_j , and decay factor α_d	Updated EAL branch current $i_d(t+1) = \alpha_d i_d(t) + \sum_{j \in \mathcal{N}_d} w_j p_j$
	<i>Soma Aggregation:</i> Inputs from EAL branch currents $i_d(t+1)$, soma decay factor β , and coupling strengths g_d	Aggregated membrane potential $\mathbf{v}(t+1) = \beta \mathbf{v}(t) + \sum_d g_d i_d(t)$
	<i>Event Generation:</i>	Event output if $\mathbf{v}(t+1) > V_{th}$, and reset potential ($\mathbf{v}(t+1) \leftarrow 0$)
Spatial Pooling Layer	Aggregated events $\mathbf{I}(x, y, t)$ from the Event Attention Layer	Event output if $\mathbf{v}(t+1) > V_{th}$, and reset potential ($\mathbf{v}(t+1) \leftarrow 0$)
FLAME Convolution Layer	Pooled event features $\mathbf{I}_{pooled}(x', y', t)$	Pooled spatio-temporal representation $\mathbf{I}_{pooled}(x', y', t)$, with reduced spatial dimensions
	<i>EA-HiPPO:</i> Event features and inter-event intervals (Δt)	Processed state $\mathbf{y}(t)$, thresholded to generate events.
	<i>NPLR Decomposition:</i> Adjusted state-space matrix \mathbf{A}_8	Adjusted state-space matrix \mathbf{A}_8 , incorporating memory retention through a decay matrix
	<i>Matrix Exponential Approximation:</i> Decomposed state-space matrix \mathbf{A}_8 , time step Δt_k	Decomposed matrix $\mathbf{A}_8 = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^* - \mathbf{P} \mathbf{Q}^*$, reducing computational complexity
	<i>FFT Convolution:</i> State vector $\mathbf{x}(t_k)$ and precomputed impulse response $\mathbf{K}(\omega)$	Approximated exponential $e^{\mathbf{A}_8 \Delta t_k}$ for efficient state updates
Layer Normalization	Intermediate activations \mathbf{x}_l from the FLAME Convolution Layer	Updated state vector $\mathbf{x}(t_{k+1})$ after efficient frequency-domain convolution
Readout Layer	Normalized features \mathbf{x}_l	Normalized activations \mathbf{x}_l , ensuring stable training by reducing variability in activations
		Final output \mathbf{y} , generated via event pooling and a linear transformation

576 C.6 Readout Layer

577 The readout layer is inspired by the *Event-SSM* architecture and employs an *event-pooling mechanism* to subsample the temporal sequence length. The pooled output is computed as $x_{pooled,k} =$
578

Algorithm 5 FLAME Convolution Layer

Require: Event train input $S(t)$, HiPPO base matrix \mathbf{A} , input coupling matrix \mathbf{B} , output coupling matrix \mathbf{C} , decay function $\mathbf{F}(\Delta t)$, time step Δt , low-rank matrices \mathbf{P} , \mathbf{Q} , total time T , rank r , state space dimension N , FFT convolution kernel $\mathbf{K}(\omega)$, threshold potential V_{th}

Ensure: Output event map $Y_{\text{event}}(t)$

Initialization

- 1: Initialize state vector $\mathbf{x} \leftarrow \mathbf{0}$ *(N -dimensional state vector)*
- 2: Initialize output $Y_{\text{event}} \leftarrow []$ *(Empty list to store event outputs)*

Precomputations

- 3: Compute event-aware HiPPO matrix: $\mathbf{A}_{\text{event}} \leftarrow \mathbf{A} \circ \mathbf{F}(\Delta t)$ *(Hadamard product with decay function)*
- 4: Perform eigendecomposition: $\mathbf{V}, \mathbf{\Lambda} \leftarrow \text{eig}(\mathbf{A}_{\text{event}})$
- 5: Decompose using NPLR: $\mathbf{A}_{\text{NPLR}} \leftarrow \mathbf{V}\mathbf{\Lambda}\mathbf{V}^* - \mathbf{P}\mathbf{Q}^*$
- 6: **for** $t = 1$ **to** T **do**

Event-Driven Dynamics

- 7: **if** $S(t)$ contains events **then**
- 8: Compute time difference: $\Delta t_k = t_{k+1} - t_k$
- 9: Approximate matrix exponential:

$$e^{\mathbf{A}_{\text{event}} \Delta t_k} \approx \mathbf{I} + \mathbf{A}_{\text{event}} \Delta t_k + \frac{(\mathbf{A}_{\text{event}})^2 (\Delta t_k)^2}{2}$$

- 10: Update state vector:

$$\mathbf{x}(t_{k+1}) \leftarrow \mathbf{x}(t_k) + \Delta t_k ((\mathbf{V}\mathbf{\Lambda}\mathbf{V}^* - \mathbf{P}\mathbf{Q}^*)\mathbf{x}(t_k) + \mathbf{B}S(t_k))$$

- 11: **else**
- 12: Update state for continuous dynamics: $\mathbf{x} \leftarrow e^{\mathbf{A}_{\text{event}} \Delta t} \mathbf{x}$
- 13: **end if**

FFT-Based Convolution for Temporal Dependencies

- 14: Transform state and kernel to frequency domain:

$$\mathbf{X}_{\text{freq}} \leftarrow \text{FFT}(\mathbf{x}), \quad \mathbf{K}_{\text{freq}} \leftarrow \text{FFT}(\mathbf{K}(\omega))$$

- 15: Perform element-wise multiplication in frequency domain:

$$\mathbf{Y}_{\text{freq}} \leftarrow \mathbf{X}_{\text{freq}} \cdot \mathbf{K}_{\text{freq}}$$

- 16: Transform back to time domain:

$$\mathbf{x}(t_{k+1}) \leftarrow \text{IFFT}(\mathbf{Y}_{\text{freq}})$$

- 17: Compute continuous output: $y_t \leftarrow \mathbf{C} \cdot \mathbf{x}(t)$
- 18: Threshold the output to generate events:

$$y_{\text{event}}(t) \leftarrow \mathbb{I}(y_t > V_{\text{th}})$$

- 19: Append $y_{\text{event}}(t)$ to Y_{event}

- 20: **end for**

Output: Y_{event} , the final event map

579 $\frac{1}{p} \sum_{i=kp}^{(k+1)p-1} x_i$, where p is the pooling factor. This operation ensures only the most relevant temporal
 580 features are retained, reducing computational burden while preserving key information. The resulting
 581 pooled sequence is passed through a linear transformation as $y = Wx_{\text{pooled}} + b$ where W and b
 582 are learnable parameters. The combination of event pooling and linear transformation provides
 583 an efficient means for deriving a final representation suitable for downstream tasks, maintaining
 584 scalability even with longer event sequences.

Algorithm 6 Readout Layer

Require: State vectors $\{x(t)\}$, pooling factor p , weights W , bias b

1: **for** each pooled time step k **do**

2: Compute pooled state:

$$x_{\text{pooled},k} \leftarrow \frac{1}{p} \sum_{i=kp}^{(k+1)p-1} x(t_i)$$

3: **end for**

4: Compute final output:

$$y \leftarrow Wx_{\text{pooled}} + b$$

5: **Output:** Model prediction y

585 C.7 FLOPs Calculation Methodology

586 The estimation of Floating Point Operations (FLOPs) per event in the FLAME model considers the
587 computational costs associated with the primary components involved in processing each event. The
588 methodology is detailed below:

1. **Event-Driven EAL Integration:** Each incoming event can trigger current updates across multiple EAL branches. The FLOPs for this stage are approximated as:

$$\text{FLOPs}_{\text{EAL}} \approx (\text{No. of branches} \times \text{No. of inputs per branch} \times 2)$$

589 This accounts for one addition and one multiplication operation per active synapse connected
590 to the EAL branches influenced by the event. Let d represent the complexity related to EAL
591 operations, so $\text{FLOPs}_{\text{EAL}} = O(d)$.

2. **EA-HiPPO State Update:** The Event-Aware HiPPO (EA-HiPPO) mechanism updates its state upon receiving input. When using a 2nd-order matrix exponential approximation and NPLR decomposition, each update involves:

- Approximately $2 \times (N \times r)$ FLOPs for the Normal-Plus-Low-Rank (NPLR) matrix-vector products, where N is the state dimension and r is the rank of the low-rank component.
- $O(N \log N)$ FLOPs for the Fast Fourier Transform (FFT) and Inverse FFT (IFFT) operations used in the convolutional implementation of the state space model.

Thus, the FLOPs for the EA-HiPPO component can be expressed as:

$$\text{FLOPs}_{\text{EA-HiPPO}} \approx 2Nr + c \cdot N \log N$$

600 where c is a small constant factor associated with the FFT/IFFT computations.

3. **Total Per-Event FLOPs:** The total computational cost per event is the sum of the FLOPs from the EAL integration and the EA-HiPPO state update:

$$\text{FLOPs}_{\text{per_event}} = \text{FLOPs}_{\text{EAL}} + \text{FLOPs}_{\text{EA-HiPPO}}$$

Substituting the approximations, we get:

$$\text{FLOPs}_{\text{per_event}} \approx O(d) + 2Nr + O(N \log N)$$

601 This methodology provides an estimate of the computational load incurred for each event processed by
602 the FLAME architecture, highlighting the contributions of its main computational blocks. The NPLR
603 decomposition and the efficiency of FFT-based convolutions are key to maintaining manageable
604 FLOP counts, especially in comparison to dense matrix operations.

605 References

- 606 [1] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.
607 *Technical report, University of Toronto*, 2009.

- [2] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7243–7252. IEEE, 2017.
- [3] Xiao Wang, Zongzhen Wu, Bo Jiang, Zhimin Bao, Lin Zhu, Guoqi Li, Yaowei Wang, and Yonghong Tian. Hardvs: Revisiting human activity recognition with dynamic vision sensors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 5615–5623, 2024.
- [4] Xiao Wang, Shiao Wang, Chuanming Tang, Lin Zhu, Bo Jiang, Yonghong Tian, and Jin Tang. Event stream-based visual object tracking: A high-resolution benchmark dataset and a novel baseline. *arXiv preprint arXiv:2408.09764*, pages 19248–19257, 2024.
- [5] Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, 2020.
- [6] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.
- [7] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.
- [8] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [9] Zhengwei Wang, Qi She, and Aljosa Smolic. Action-net: Multipath excitation for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13214–13223, 2021.
- [10] Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. Tam: Temporal adaptive module for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13708–13718, 2021.
- [11] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3202–3211, 2022.
- [12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.
- [13] Jiazhou Zhou, Xu Zheng, Yuanhuiyi Lyu, and Lin Wang. Exact: Language-guided conceptual reasoning and uncertainty estimation for event-based action recognition and more. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18633–18643, 2024.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.
- [16] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Gate-shift-fuse for video action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10913–10928, 2023.
- [17] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021.

- [18] Zhen Xing, Qi Dai, Han Hu, Jingjing Chen, Zuxuan Wu, and Yu-Gang Jiang. Svformer: Semi-supervised video transformer for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18816–18826, 2023.
- [19] Lan Chen, Dong Li, Xiao Wang, Pengpeng Shao, Wei Zhang, Yaowei Wang, Yonghong Tian, and Jin Tang. Retain, blend, and exchange: A quality-aware spatial-stereo fusion approach for event stream recognition. *arXiv preprint arXiv:2406.18845*, 2024.
- [20] Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv preprint arXiv:2403.02308*, 2024.
- [21] Lianghai Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.
- [22] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *ArXiv*, abs/2401.10166, 2024.
- [23] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Video-mamba: State space model for efficient video understanding. *arXiv preprint arXiv:2403.06977*, 2024.
- [24] Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, et al. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4):e2109194119, 2022.
- [25] Nicolas Perez-Nieves, Vincent CH Leung, Pier Luigi Dragotti, and Dan FM Goodman. Neural heterogeneity promotes robust learning. *Nature communications*, 12(1):5791, 2021.
- [26] Julian Rossbroich, Julia Gygax, and Friedemann Zenke. Fluctuation-driven initialization for spiking neural network training. *Neuromorphic Computing and Engineering*, 2(4):044016, 2022.
- [27] Bojian Yin, Federico Corradi, and Sander M Bohte. Accurate online training of dynamical spiking neural networks through forward propagation through time. *arXiv preprint arXiv:2112.11231*, 2021.
- [28] Biswadeep Chakraborty and Saibal Mukhopadhyay. Heterogeneous neuronal and synaptic dynamics for spike-efficient unsupervised learning: Theory and design principles. In *The Eleventh International Conference on Learning Representations*, 2023.
- [29] Hanle Zheng, Zhong Zheng, Rui Hu, Bo Xiao, Yujie Wu, Fangwen Yu, Xue Liu, Guoqi Li, and Lei Deng. Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics. *Nature Communications*, 15(1):277, 2024.
- [30] Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In *International Conference on Machine Learning*, pages 35798–35816. PMLR, 2023.
- [31] Ilyass Hammouamri, Ismail Khalfaoui-Hassani, and Timothée Masquelier. Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In *The Twelfth International Conference on Learning Representations*, 2024.
- [32] Sicheng Shen, Dongcheng Zhao, Guobin Shen, and Yi Zeng. Tim: An efficient temporal interaction module for spiking transformer. *arXiv preprint arXiv:2401.11687*, 2024.
- [33] Shimin Zhang, Qu Yang, Chenxiang Ma, Jibin Wu, Haizhou Li, and Kay Chen Tan. Tc-lif: A two-compartment spiking neuron model for long-term sequential modelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16838–16847, 2024.

- [34] Amirreza Yousefzadeh, Mina A Khoei, Sahar Hosseini, Priscila Holanda, Sam Leroux, Orlando Moreira, Jonathan Tapson, Bart Dhoedt, Pieter Simoens, Teresa Serrano-Gotarredona, et al. Asynchronous spiking neurons, the natural key to exploit temporal sparsity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(4):668–678, 2019.
- [35] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *arXiv preprint arXiv:2210.04195*, 35:20717–20730, 2022.
- [36] Anand Subramoney. Efficient real time recurrent learning through combined activity and parameter sparsity. *arXiv preprint arXiv:2303.05641*, 2023.
- [37] Xueyuan She, Saurabh Dash, and Saibal Mukhopadhyay. Sequence approximation using feedforward spiking neural network for spatiotemporal learning: Theory and optimization methods. In *International Conference on Learning Representations*, 2021.
- [38] Chang Liu, Xiaojuan Qi, Edmund Y Lam, and Ngai Wong. Fast classification and action recognition with event-based imaging. *IEEE access*, 10:55638–55649, 2022.
- [39] Biswadeep Chakraborty and Saibal Mukhopadhyay. Heterogeneous recurrent spiking neural network for spatio-temporal classification. *arXiv preprint arXiv:2211.04297*, 17:994517, 2022.
- [40] Carmen Martin Turrero, Maxence Bouvier, Manuel Breitenstein, Pietro Zanuttigh, and Vincent Parret. Alert-transformer: Bridging asynchronous and synchronous machine learning for real-time event-based spatio-temporal data. *arXiv preprint arXiv:2402.01393*, 2024.
- [41] Mark Schöne, Neeraj Mohan Sushma, Jingyue Zhuge, Christian Mayr, Anand Subramoney, and David Kappel. Scalable event-by-event processing of neuromorphic sensory signals with deep state-space models. *Neuromorphic Computing Conference*, 2024.
- [42] Simone Undri Innocenti, Federico Becattini, Federico Pernici, and Alberto Del Bimbo. Temporal binary representation for event-based action recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10426–10432. IEEE, 2021.
- [43] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatz, and Yiannis Andreopoulos. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. *IEEE Transactions on Image Processing*, 29:9084–9098, 2020.
- [44] Yongjian Deng, Hao Chen, Hai Liu, and Youfu Li. A voxel graph cnn for object classification with event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1172–1181, 2022.
- [45] Shu Miao, Guang Chen, Xiangyu Ning, Yang Zi, Kejia Ren, Zhenshan Bing, and Alois Knoll. Neuromorphic vision datasets for pedestrian detection, action recognition, and fall detection. *Frontiers in neurorobotics*, 13:38, 2019.
- [46] Qinyi Wang, Yexin Zhang, Junsong Yuan, and Yilong Lu. Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1826–1835. IEEE, 2019.
- [47] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2661–2671, 2021.
- [48] Yansong Peng, Yueyi Zhang, Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. Get: Group event transformer for event-based vision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6038–6048, 2023.
- [49] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022.

- 751 [50] Hongwei Ren, Yue Zhou, Jiadong Zhu, Haotian Fu, Yulong Huang, Xiaopeng Lin, Yuetong Fang,
752 Fei Ma, Hao Yu, and Bojun Cheng. Rethinking efficient and effective point-based networks for
753 event camera classification and regression: Eventmamba. *arXiv preprint arXiv:2405.06116*,
754 2024.
- 755 [51] Lin Zuo, Yongqi Ding, Wenwei Luo, Mengmeng Jing, Xianlong Tian, and Kunshan Yang.
756 Temporal reversed training for spiking neural networks with generalized spatio-temporal repre-
757 sentation. *arXiv preprint arXiv:2408.09108*, 2024.
- 758 [52] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-
759 driven transformer. *Advances in neural information processing systems*, 36, 2024.