# Efficient Neuromorphic Data Processing: A Unified Framework for Pipeline Optimization and Lightweight Preprocessing*

*Note: Sub-titles are not captured in Xplore and should not be used


1st Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

2nd Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

3rd Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

4th Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

5th Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

6th Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID



***Abstract*—To address the computational efficiency bottleneck in real-time processing of neuromorphic data (e.g., DVS128 Gesture), this paper proposes an end-to-end acceleration solution that integrates data pipeline optimization and lightweight preprocessing. At the system level, the use of `num_workers` for parallel loading and `prefetch_factor` for prefetching significantly enhances the throughput of the data pipeline. At the algorithmic level, a dynamic time window event frame aggregation method based on spatiotemporal sparsity is introduced, along with a low-computation pulse noise filtering module leveraging local spatiotemporal correlations. Additionally, structured model pruning is employed to reduce redundant connections and lower computational overhead. Experimental results demonstrate that on an NVIDIA 4070 GPU, the inference speed on the test set is doubled, while achieving classification accuracies of 92.36% (test set) and 98.97% (training set). This work combines data pipeline optimization with lightweight preprocessing, providing a high real-time solution for neuromorphic data processing in edge computing scenarios, with significant practical engineering value.**



***Index Terms*—Neuromorphic Computing, Data Pipeline Optimization, Pulse Noise Filtering**


## I. Introduction

Spiking Neural Networks(SNN),which are regarded as the third generation of neural network models,offer a more biologically plausible simulation of neural dynamic by incorporating not only neuronal and synaptic states but also temporal information into their computational framework. [1], [2]Unlike traditional artificial neural networks (ANNs),which relay on continuous-valued activations,SNNs process information through discrete spike events,letting them to capture the temporal dependencies inherent in biological systems.This temporal sensivity,combined with energy-efficient computation,allows SNNs to become as a promising alternative for modeling complex cognitive tasks with potentially superior performance and lower power consumption. [3]

Neuromorphic data, characterized by its event-driven nature and sparse sensory representation, has emerged as a compelling paradigm across various real-time application domains, including gesture recognition, object tracking, and autonomous navigation. Neuromorphic data, characterized by its event-driven nature and sparse sensory representation, has emerged as a compelling paradigm across various real-time application domains, including gesture recognition, object tracking, and autonomous navigation.However, the efficient processing of such data remains a significant challenge, particularly in edge computing scenarios where computational resources are limited. A critical bottleneck lies in the substantial latency introduced by data loading and preprocessing. Traditional approaches to address this issue often focus on either algorithmic optimizations or hardware acceleration in isolation, failing to fully exploit the synergistic potential of a unified framework. [4]

This paper proposes a novel solution that seamlessly integrates data pipeline optimization with lightweight preprocessing techniques to achieve end-to-end acceleration. At the system level, we leverage parallel loading and prefetching mechanisms to maximize data throughput. On the algorithmic front, we introduce a computationally efficient pulse noise filtering module based on local spatiotemporal correlations.This module enhances the robustness of our model against noise while preserving critical temporal dynamics. Additionally, we incorporate a self-attention mechanism to augment the


Identify applicable funding agency here. If none, delete this.

model's understanding of feature channel interdependencies, thereby improving its representational capacity. [5] To further enhance computational efficiency, we employ structured model pruning,which systematically reduces redundant connections. [6]

Our contributions are threefold:

(1)we compared the effects on accuracy and training speed when setting different frame numbers in the ddateset.

(2)we proposed a lightweight preprocessing method that combines pulse noise filtering and self-attention mechanisms based on the network proposed by WeiFang [7]to enhance the model's robustness and representational capacity.Meanwhile we conducted experiments on the DVS-Gesture dataset and achieved experimental results.

(3)we introduced structured model pruning to reduce redundant connections, thereby improving computational efficiency.

## II. Related Work

### A. Spiking Neural Networks

Spiking Neural Networks(SNNs) mimics the pulse emission characteristics of biological neurons, drives computing with discrete events, and has significant advantages in energy-sensitive scenarios (such as edge devices). [3] A major research focus in the field of SNNs is the conversion from artificial neural networks(ANNs) to SNNs [8],which is dedicated to addressing the challenges of accuracy degradation and training instability during the conversion process. [9] Zhang et al. (2025) proposed a method called STNN-SNN, a Spatial-Temporal Attention Aggregator SNN framework,which can dynamically attend to and capture dependencies between spatial and temporal domains. [10]

### B. Spatial-Temporal-Polarity Coherence Filtering

In event-based neuromorphic data processing, spatio-temporal polarity coherent filtering is an effective denoising processing method. Unlike conventional frame-based filtering, this method leverages the unique characteristics of event streams, where each event is defined by its spatial coordinates, timestamp, and polarity. Spatio-temporal polarity coherent filtering has been widely applied in event-based neuromorphic vision tasks. Sironi et al.(2018) used polarity and spatial-temporal neighborhood statistical characteristics to robustly classify events. [11]

### C. Attention Mechanism of SNNs

Attention mechanism has attracted considerable attention since its inception and has been widely used in various neural network architectures.Xue et al. (2024) introduced Gated Attention Coding (GAC) [12], a plug-and-play module that encodes inputs into powerful representations.Yu et al. (2024) proposed a Frequency-based Spatial-Temporal Attention(FSTA) model to enhance feature learning in SNNs. [13] Shen et al.(2024) innovatively implemented an attention mechanism in SNNs called Temporal Interaction Module (TIM),which was designed to augment the temporal data processing abilities within SNN architectures. [14]

## III. Methodology

in this section, we first introduce the fundamental computational units of SNNs,namely the most widely used spiking neuron:the Leaky Integrate-and-Fire (LIF) neuron [15].Then,we discussed the polarity coherence filtering method used for dataset denoising. At last, we present the model improved with Squeeze-and-Excitation Block (SEBlock). [5]

### A. LIF neuron

An LIF neuron is depicted by the following equations:

$$\tau \frac{dV(t)}{dt} = -\left(V(t) - V_{\text{reset}}\right) + I(t) \tag{1}$$

where $V(t)$ denotes the membrance potential of neuron at time $t$,$I(t)$ represents the input to neuron at time $t$, $V_{\text{reset}}$ is the reset potential, and $\tau$ is the time constant of the neuron. When the membrane potential $V(t)$ goes beyond the certain threshold $V_{\text{thresh}}$, the neuron will emit a spike and reset its membrane potential to $V_{\text{reset}}$. The output spike can be defined as:

$$S(t) = \begin{cases} 1, & \text{if } V(t) \geq V_{\text{thresh}} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $S(t)$ is the output spike at time $t$. The LIF neuron model captures the essential dynamics of biological neurons, which includes the integration of inputs over time and the generation of discrete spikes based on membrane potential thresholds.

### B. Spatio-temporal polarity coherence filtering: formula analysis

Given an event stream $\mathcal{E} = \{e_i\}_{i=1}^{N}$, where each event $e_i = (t_i, x_i, y_i, p_i)$ consists of timestamp $t_i$, spatial coordinates $(x_i, y_i)$, and polarity $p_i \in \{0, 1\}$, the processing consists of two main steps: polarity-consistent filtering and temporal frame integration.

*1) Polarity-Consistent Spatio-Temporal Filtering:* For each event $e_i$, define its feature vector as:

$$\mathbf{f}_i = \left( \left\lfloor \frac{t_i}{\Delta t} \right\rfloor, x_i, y_i \right)$$

where $\Delta t$ is the time window (e.g., 1000).

Construct a $k$-d tree using all $\mathbf{f}_i$ and, for each event, find its neighbors within radius $r$:

$$\mathcal{N}_i = \{ j \mid \|\mathbf{f}_j - \mathbf{f}_i\|_2 \leq r \}$$

An event $e_i$ is retained if the number of neighbors with the same polarity exceeds a threshold $\theta$:

$$\sum_{j \in \mathcal{N}_i} \mathbb{I}(p_j = p_i) > \theta$$

where $\mathbb{I}(\cdot)$ is the indicator function.

*2) Temporal Frame Integration (16 Frames Example):* Let the filtered events be $\{e_k\}_{k=1}^{N'}$ with timestamps $t_k$. Define $t_{\text{start}} = t_1$, $t_{\text{end}} = t_{N'}$, and total duration $T = t_{\text{end}} - t_{\text{start}}$.

Set the base frame width and remainder as:

$$d = \left\lfloor \frac{T}{16} \right\rfloor, \quad r = T \bmod 16$$

For frame $i$ ($i = 0, 1, \ldots, 15$), the time interval is:

$$t_{\text{low}}^{(i)} = t_{\text{start}} + i \cdot d + \min(i, r) \tag{3}$$

$$t_{\text{high}}^{(i)} = t_{\text{low}}^{(i)} + d + \delta_i \tag{4}$$

where

$$\delta_i = \begin{cases} 1, & i < r \\ 0, & i \geq r \end{cases} \tag{5}$$

Each frame $F_i$ is constructed by accumulating events in $[t_{\text{low}}^{(i)}, t_{\text{high}}^{(i)})$:

$$F_i(c, x, y) = \sum_{k \in \mathcal{S}_i} \mathbb{I}(x_k = x, y_k = y, p_k = c)$$

where $\mathcal{S}_i = \{k \mid t_{\text{low}}^{(i)} \leq t_k < t_{\text{high}}^{(i)}\}$ and $c \in \{0, 1\}$ denotes the polarity channel.

The final output is a tensor of shape $16 \times 2 \times H \times W$.

### C. The model improved with Squeeze-and-Excitation Block (SEBlock)

when the filtering process for the dataset is completed, we can use the processed dataset to train the model. In this paper we proposed a model based on the WeiFang's model [7] and improved it with Squeeze-and-Excitation Block (SEBlock) [5],alongside we conducted various experiments on this model and performed detailed comparisons.The model architecture is shown in Fig 1 and more Specific details will be presented in the following sections.

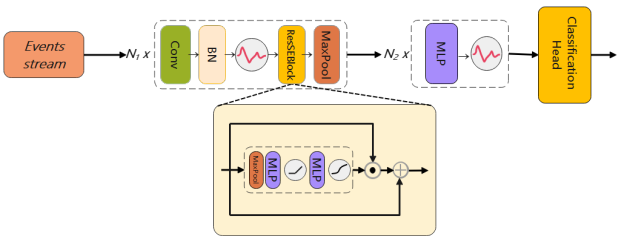



Fig. 1: The Model Architecture with Squeeze-and-Excitation Block (SEBlock)

## IV. Experiments

### A. Experimental Setup

**Dataset Description** In this experiment, we utilize the DVS Gesture dataset, a publicly available dataset for gesture recognition based on Dynamic Vision Sensor (DVS). The dataset comprises 1,342 samples, each corresponding to one of 11 distinct gesture categories. Each gesture is performed multiple times by approximately 29 different subjects. Each sample is stored in the form of an event stream. The dataset also provides training and test set splits, facilitating model evaluation on real-world event data. The data was collected under varying lighting and motion conditions to enhance diversity.

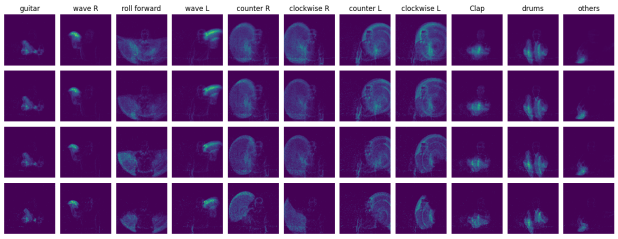



Fig. 2: DVS Gesture Dataset in Different Frame Numbers

**Evaluation Metrics** To comprehensively evaluate model performance, we employ five key metrics: (1) Training speed (samples/sec), measuring computational efficiency by tracking processed samples per second during training; (2) Training accuracy (%), monitoring the model's learning progress by calculating correct predictions on training data; (3) Test accuracy (%), the primary classification metric evaluated on held-out test samples; (4) Training loss, recorded as an observational metric to analyze model behavior during training; (5) Test loss, evaluating generalization capability on held-out data.

All experiments were conducted on a single NVIDIA RTX 4070 GPU with a batch size of 16, using the Adam optimizer, Automatic Mixed Precision (AMP) for faster computation, and the CuPy library for GPU-accelerated array operations. This multi-faceted evaluation captures both computational efficiency (training speed) and model effectiveness (accuracy/loss), while the training loss serves purely as a diagnostic indicator rather than a control signal.

### B. Impact of Frame Count on Model Performance

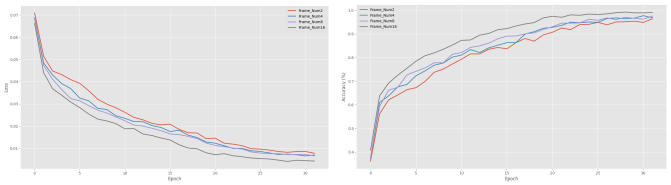
When processing DVS Gesture data, it is often necessary to convert the raw event stream into some form of image representation (e.g., event frames, accumulated frames, or time surfaces) so that it can be input into a convolutional neural network (CNN) for classification or other tasks. In this process, frame number is a key parameter, which indicates how many time windows the entire event stream should be divided into. Within each window, an "event frame" is generated as an input feature.

In this experiment, we process the DVS Gesture dataset on different frame numbers, as shown in Fig 2. Each row represents the processed dataset with different frame numbers, from top to bottom the number of frames is 2, 4, 8, and 16. And each column represents a different category, from left to right, which is guitar, right hand wave, forearm roll forward, left hand wave, right hand counter clockwise, right hand clockwise, left hand counter clockwise, left hand clockwise, clap, drums and others.

Through the sample of dataset, we can see that for most categories there is only a change in brightness caused by the difference of frame number, but for certain categories, such

as the counter clockwise and clockwise rotation of the left and right hands, it is difficult to distinguish the corresponding images in the dataset with a small number of frames.

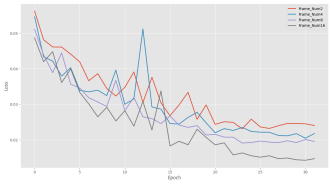
To detect the influence of frame number on the model training, we use the DVS Gesture dataset processed with different frame numbers to train the basic model and compare the performance of the model. The corresponding training results are shown in Fig3. The results show that the model with more



(a) Train Loss Curve



(b) Train Accuracy Curve



(c) Test Loss Curve

(d) Test Accuracy Curve

Fig. 3: Performance of Base Model at Different Frame Numbers

frames can achieve higher accuracy. The Fig3 demonstrate that the frame partitioning strategy exhibits negligible impact on both model loss and accuracy during the initial 10 training epochs. However, subsequent training reveals a significant performance divergence - higher frame counts consistently yield superior results, with the baseline model achieving lower loss values and higher accuracy when processing certain movement sampled frames. This phenomenon can be attributed to the increased information density afforded by finer temporal segmentation, which enables more comprehensive feature extraction throughout the network's deeper layers.

In addition to loss and accuracy, training speed is also an important metric for evaluating model performance. The Fig4 below illustrates the relationship between Average training speed and Maximum test accuracy during model training across different frame numbers.

As shown in Fig4, reducing the number of frames per event significantly accelerates the training speed. However, fewer frames lead to a certain degree of information loss, which consequently reduces the accuracy. In this experiment, the highest accuracy of 91.0% is achieved when each event is divided into 16 frames, while the training speed is the slowest at 7.5. Conversely, when each event is divided into 2 frames, the accuracy drops to its lowest at 83.0%, but the training speed increases nearly tenfold compared to the slowest speed.

### C. Impact of Denoising on Model Efficacy

To further enhance the model's training performance, we performed denoising on the dataset based on setting the

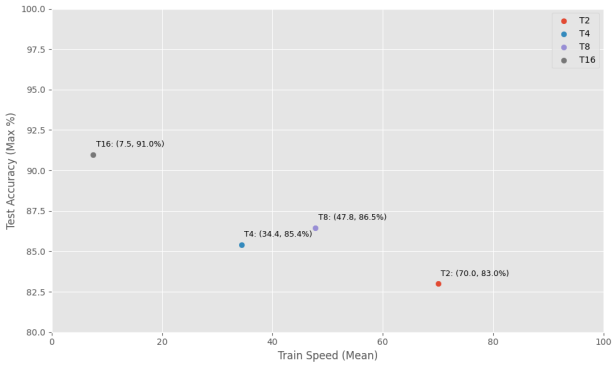



Fig. 4: Speed-Accuracy Plot

number of frames for event segmentation to 16, aiming to achieve superior training outcomes.We applied the spatiotemporal polarity consistency filtering method, as introduced in the Methodology chapter, to process the dataset. The frame

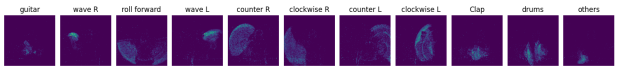



Fig. 5: Denoised Dataset

of each category from the processed dataset are illustrated in Fig8. The denoised images exhibit a high level of clarity and smoothness, with minimal visual artifacts. Essential details, such as edges and textures, are well-preserved, ensuring the integrity of the original data. The dataset presents a clean and visually consistent representation, characterized by a uniform and noise-free appearance. The overall visual quality of the denoised images is both natural and interpretable, making them suitable for subsequent analysis.

To further optimize based on the results of the previous experiment, we similarly segmented each event stream in the denoised dataset into 16 frames and trained the baseline model for 32 epochs to compare the training performance. The transformations of key parameters during the training process are shown in Fig6. By analyzing Fig6a and Fig6b, which respectively depict the training loss and training speed of basic model on both the original dataset and the denoised dataset, it can be observed that except for the slower training speed of the base model in the first epoch on the denoised dataset compared to the original dataset, the performance of the two datasets during the training process is similar in all other cases. However, in the test process,

### D. Assessing Model Performance with Attention Mechanism

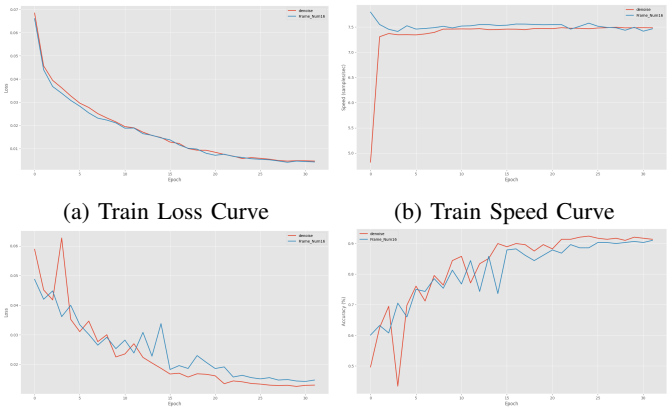
## V. Conclusion

## References

## References

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

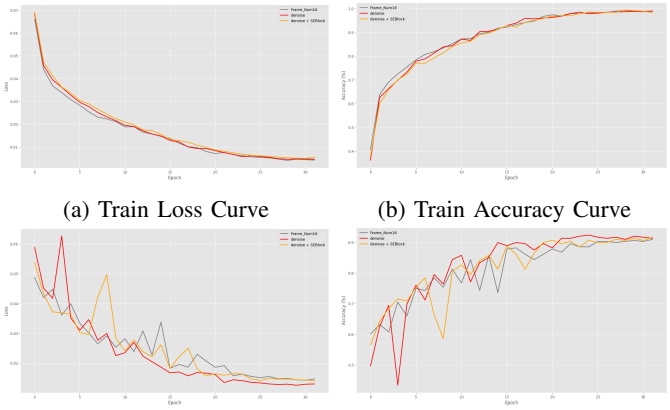
[2] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. USA: Cambridge University Press, 2014.



(a) Train Loss Curve (b) Train Speed Curve

(c) Test Loss Curve (d) Test Accuracy Curve

Fig. 6: Performance of Base Model at



(a) Train Loss Curve (b) Train Accuracy Curve

(c) Test Loss Curve (d) Test Accuracy Curve

Fig. 7: Performance of Base Model at Different Frame Numbers

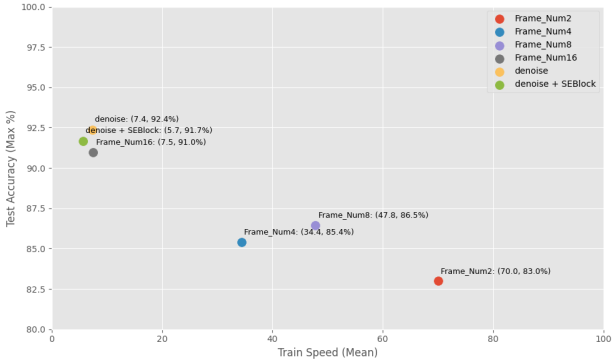



Fig. 8: Speed-Accuracy Plot of All Training Processes


[3] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[4] Y. Ding, L. Zuo, M. Jing, P. He, and Y. Xiao, "Shrinking your timestep: Towards low-latency neuromorphic object recognition with spiking neural network," 2024.

[5] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2019.

[6] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," 2015.

[7] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," 2021.

[8] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," in *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 677–687.

[9] B. Han, G. Srinivasan, and K. Roy, "Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 555–13 564.

[10] T. Zhang, K. Yu, X. Zhong, H. Wang, Q. Xu, and Q. Zhang, "Staa-snn: Spatial-temporal attention aggregator for spiking neural networks," 2025. [Online]. Available: https://arxiv.org/abs/2503.02689

[11] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1731–1740.

[12] X. Qiu, R.-J. Zhu, Y. Chou, Z. Wang, L. jian Deng, and G. Li, "Gated attention coding for training high-performance and efficient spiking neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2308.06582

[13] K. Yu, T. Zhang, H. Wang, and Q. Xu, "Fsta-snn:frequency-based spatial-temporal attention module for spiking neural networks," 2025. [Online]. Available: https://arxiv.org/abs/2501.14744

[14] S. Shen, D. Zhao, G. Shen, and Y. Zeng, "Tim: An efficient temporal interaction module for spiking transformer," 2024. [Online]. Available: https://arxiv.org/abs/2401.11687

[15] N. Brunel and M. C. Rossum, "Quantitative investigations of electrical nerve excitation treated as polarization," *Biol. Cybern.*, vol. 97, no. 5–6, p. 341–349, Dec. 2007.