Spiking vs. Polynomial Kernels for Gesture Recognition

Ginevra Bozza, Saijal Singhal

***Abstract*—We compare two fundamentally different models for event-based gesture recognition on the DVS128 dataset: PLEIADES, which uses causal polynomial kernels for frame-wise predictions, and S-TLLR, a spiking neural network trained with a local, biologically inspired learning rule. PLEIADES achieves perfect accuracy with fast, low-latency responses, while S-TLLR, though slightly less accurate overall, demonstrates surprisingly strong early performance. Our analysis of latency vs. accuracy reveals key trade-offs between responsiveness and biological plausibility, informing the design of efficient, real-time gesture recognition systems.**

***Index Terms*—Orthogonal polynomials, event-based, spiking neural networks, gesture recognition, deep learning**

I. Introduction

Gesture recognition is the task of identifying and interpreting human gestures, such as hand waves, swipes, or poses, as input for computational systems bypassing the need for physical touch or traditional input devices. Examples of gesture recognition can be seen in hand gestures to navigate in virtual reality environments [1], sign language recognition for accessibility tools [2] [3], etc. Gesture recognition lies at the intersection of computer vision, machine learning, and human-computer interaction, requiring systems to not only detect but also classify motion patterns that vary between individuals and contexts.

The complexity of gesture recognition arises from the inherently dynamic nature of gestures. Unlike static image classification, gestures evolve over time and must be interpreted as temporal sequences. This adds layers of difficulty such as accounting for varying gesture speeds, occlusions, and inter-user variability. Moreover, recognizing a gesture accurately often requires the system to reason about both spatial configurations (e.g., hand shape) and temporal dynamics (e.g., how the hand moves through space). As a result, gesture recognition systems must be capable of modeling spatial and temporal features robustly and efficiently.

A. Temporal Kernels

Temporal kernels are specialized filters that are used to model how information evolves over time in sequential data. In the context of gesture recognition, they are applied to sequences of visual input (e.g., frames or events) to capture temporal dependencies, such as motion direction or velocity. A temporal kernel can be thought of as a sliding window over the time axis, enabling a model to aggregate and process information from multiple time steps. This is particularly useful in gesture recognition where the meaning of a gesture often depends on the sequence and timing of motions, not just the current spatial configuration.

Different models implement temporal kernels in various forms. For instance, 3D convolutional networks extend standard 2D convolutions by incorporating a temporal dimension, enabling the model to learn spatiotemporal features directly. More advanced methods, like PLEIADES [4], replace standard convolutional kernels with mathematically produced temporal kernels, which allow for a more compact and expressive representation of temporal dynamics. These kernels are fully causal—meaning they depend only on past input—and can be designed to operate efficiently in real time. By effectively modeling how gestures unfold over time, temporal kernels help improve the robustness and responsiveness of gesture recognition systems, especially in applications requiring fast decision-making.

B. Event-based Data

Event-based data refers to visual information captured as a continuous stream of asynchronous events, rather than traditional full image frames. These events are generated whenever a change in brightness occurs at a specific pixel location and time. This sparse and temporally precise data representation significantly reduces redundancy and power consumption, making it ideal for real-time vision tasks. Unlike conventional cameras that operate at fixed frame rates, event-based sensors only report dynamic changes, offering high temporal resolution with low latency.

The DVS128 dataset, a widely used benchmark in this domain, was introduced as part of IBM's research into efficient gesture recognition systems [5]. It was collected using a Dynamic Vision Sensor (DVS) and includes recordings of various hand gestures performed under different lighting conditions. Originally developed for a low-power, event-driven recognition pipeline using neuromorphic hardware, this dataset has since become a standard for evaluating event-based gesture recognition algorithms due to its realistic and diverse scenarios.

Gesture recognition is a natural fit for event-based data, as gestures are inherently defined by motion. The asynchronous, high-speed nature of event streams allows models to detect subtle movements more quickly and with less computational overhead than frame-based systems. However, event-based data also introduces new challenges: it is sparse, lacks absolute intensity information, and requires models to process inputs in a temporally-aware manner. This has led to the development of specialized architectures—such as spiking neural networks and temporally-convolved models—that are capable of leveraging the temporal richness of event streams. By aligning with the properties of gesture dynamics, event-based data not only enables faster and more efficient recognition but also opens new possibilities for low-latency, real-time interaction systems.

Recognizing gestures quickly and accurately from event-based data is a challenging problem. Unlike traditional video, event cameras produce streams of sparse, asynchronous signals that require models to make sense of motion and timing without full frames. The key challenge is building systems that can respond fast, ideally in real time, while still maintaining high accuracy. In this work, we explore how different neural architectures handle this trade-off between speed and accuracy in event-based gesture recognition.

II. Literature Review

Traditional computer vision techniques for gesture recognition are based primarily on video-based data, where sequences of RGB or depth frames are analyzed over time. Early approaches used hand-made features such as optical flow [6], motion history images (MHI) [7], and histogram-based descriptors to capture temporal patterns [8]. These were then combined with classifiers like support vector machines (SVMs) [9] or hidden Markov models (HMMs) [10]. Although effective in constrained settings, these methods struggled to generalize in varying lighting conditions, backgrounds, and gesture execution styles.

With the rise of deep learning, Convolutional Neural Networks (CNNs) and their extensions became the dominant tools for gesture recognition [11] [12]. 3D CNNs, which apply convolutions in both spatial and temporal dimensions, have been widely adopted to capture motion across consecutive frames [13]. Recurrent neural networks (RNNs) and long short-term memory (LSTM) units are also commonly used to model temporal dependencies in video sequences [14]. More recently, attention-based models like transformers have been employed to learn long-range temporal correlations more effectively [15] [16]. These models achieve high accuracy, but often require large datasets, substantial compute resources, and consistent frame-rate input, factors that limit their suitability for edge devices or real-time applications [17].

A. Parameterization of Temporal kernels and spatiotemporal network

Recent gesture-recognition studies have moved away from fully-learned, dense temporal filters toward compact parameterizations that cut memory and compute without sacrificing temporal expressiveness. PLEIADES [4] introduced the most aggressive approach to date, expressing every 1-D kernel as a small set of coefficients that mix fixed orthogonal Jacobi polynomials; this shrinks a 10-tap filter to five learnable numbers and lets the same weights be re-sampled to different bin sizes on the fly, a key advantage for low-latency, event-driven inference. Video-based work explored lighter alternatives like temporal-Shift Modules which replaced convolutions with channel shifts to achieve real-time hand-gesture demos on Jetson Nano with almost zero additional parameters [18], while depth- and channel-separable 1-D filters were embedded in transformer backbones for sparse-EMG and vision streams to reduce multiply–accumulate counts by an order of magnitude [19]. Partial 3-D convolution further pruned kernel weights by convolving only a subset of channels, cutting latency for resource-aware gesture systems [20]. For event data, Sironi-style exponential–decay kernels were refined in 2023 by a two-stage temporal-convolution network that recognised DVS gestures at 97.7% accuracy while running at 4.7mW [21].

B. Spiking Neural Networks

Spiking Neural Networks (SNNs) are characterized by their ability to process information through discrete spikes, similarly to biological neurons, making them well-suited for processing asynchronous event-based data. However, training deep SNNs presents significant challenges, due to the non-differentiability of the spike events and a substantial computational complexity. Traditional methods like Backpropagation Through Time (BPTT) have been adapted for SNNs, but often result in high computational demand and require extensive memory, as the depth of the network increases. To overcome these issues, biologically inspired learning rules have been proposed, such as Spike-Timing Dependent Plasticity (STDP), an unsupervised learning rule, which enables the network to adapt based on temporal correlations in the input data, lacking, however, scalability for deeper networks.

Building upon this, the S-TLLR (STDP-inspired Temporal Local Learning Rule) [22] introduces a three-factor learning mechanism, that combines local spike timing with a top-down learning signal. This approach enhances generalization, supports online learning, and reduces memory complexity, facilitating the training of deep SNNs and achieving performance comparable to traditional backpropagation through time (BPTT) on event-based datasets, including gesture recognition tasks.

Recent advancements in SNNs architectures aim to enhance their ability to model complex spatio-temporal patterns, which is crucial in event-based gesture recognition. To better handle the temporal complexity of gestures, architectures like Spike Gating Flow (SGF) [23] have introduced hierarchical processing units, each including a feature extraction layer, an event-driven layer, and a histogram-based training layer, and it achieved an 87.5% on the DVS128 Gesture dataset with only a single training epoch, demonstrating efficiency in few-shots learning scenarios. Similarly, Spatio-Temporal Synaptic Connection SNN (STSC-SNN) [24], extendend this approach by incorporating convolutions and attention mechanisms to expand the spatiotemporal receptive fields of neurons, improving the network's ability to model temporal correlations in gesture sequences recorded by event cameras. Yao et al. (2022) [25] proposed Multi-dimentional Attention SNN (MA-SNN), integrating attention mechanism across temporal, channel and spatial dimensions, optimizing membrane potentials based on attention weights, which results in reduced spiking activity and improved energy efficiency. Further advancements include the Temporal-Guided Spiking Neural Network (TG-SNN) [26], which improves the modeling of long-range temporal relationships in gesture recognition, by using temporal segmentation and 3D convolutions to efficiently extract spatiotemporal features.

By integrating attention, temporal abstraction, and hierarchical processing, modern SNNs are more effective in interpreting the sparse, high-temporal resolution data generated by event-based cameras, being increasingly aligned with the practical demands of gesture recognition tasks in real-world environments.

C. Event-based data

Event-based cameras - such as the Dynamic Vision Sensor (DVS) used in the DVS128 Gesture Dataset [5] - produce sparse and asynchronous data streams, that capture changes in luminance with high temporal resolution and low-latency. Unlike frame-based cameras that capture full images at fixed intervals, event-based cameras record changes in brightness at each pixel, resulting in a stream of events characterized by their spatial coordinates, timestamp, and polarity [27]. These properties are ideal for applications requiring rapid motion detection and dynamic scene analysis, but the effective processing of this data requires specialized techniques to decide how to aggregate events and where to learn temporal structure. One common approach is to convert the asynchronous event streams into synchronous frames, using binning techniques. Once events are synchronized this way, it is possible to apply conventional convolutions [4]. Another solution is to keep the data fully asynchronous. Deep SNNs trained with local plasticity update weights at each event, reducing memory complexity from O(T n) to O(n), and match BPTT accuracy on gesture benchmark, while learning online [22]. Recent advancements have also explored hybrid architectures that combine the strenght of both frame-based and event-driven processing, like hybrid SNN-ANN architectures, integrating initial spiking layers to capture temporal information, followed by traditional artificial neural network layers for classification [28].

III. Methodology

A. TENNs-PLEIADES model architecture

The TENNs-PLEIADES model is a lightweight yet high-performing spatiotemporal neural network designed for real-time gesture recognition on event-based data. It is particularly optimized for use with the DVS128 sensor, which outputs asynchronous streams of events rather than full image frames. To handle such input effectively, PLEIADES processes incoming data as 10 ms binned event frames and passes them through a five-stage spatiotemporal backbone. Each stage, or block, is divided into a temporal half followed by a spatial half, mimicking the (1+2)D convolution factorization used in R(2+1)D networks. This design enables efficient separation of temporal and spatial reasoning, while avoiding the computational cost of full 3D convolutions.

The overall structure of the network is illustrated in Figure 1, which depicts the full processing pipeline from the event stream to gesture prediction. Each spatiotemporal block begins with a temporal convolution (shown in blue), followed by a spatial convolution (shown in orange/yellow). The darker blocks represent standard full convolutions, while the lighter ones indicate depthwise-separable (DWS) convolutions—used in later layers to reduce computational cost. The temporal kernels all have a shape of (10×1×1), meaning that the model observes 10 frames at a time per pixel, while the spatial kernels are typically (1×3×3), capturing neighborhood context in each frame. Output channels increase progressively from 8 to 256 through the backbone, and stride-2 spatial convolutions downsample the input resolution step-by-step from 128×128 down to 4×4.

In the temporal half, each pixel's time series is convolved using a 1D kernel of size 10, capturing fine-grained temporal dynamics. Rather than learning individual weights, PLEIADES parameterizes each temporal kernel using a Jacobi polynomial basis ($\alpha = \beta = -0.25$, degree ≤ 4), meaning only five coefficients are learned per channel pair. This polynomial formulation drastically reduces the parameter count and supports temporal kernel re-sampling for different bin sizes without retraining. Depending on the block, these temporal convolutions may be standard or depthwise-separable, where one compressed kernel is applied per input channel followed by a 1×1 pointwise convolution. Causal Group Normalization (with 4 groups) is applied to each temporal output, ensuring that the network uses only past and present frames, thereby preserving strict online causality. A ReLU activation follows, chosen for its simplicity and the sparsity it induces in the activation map.

In the spatial half, the temporally processed frame under-

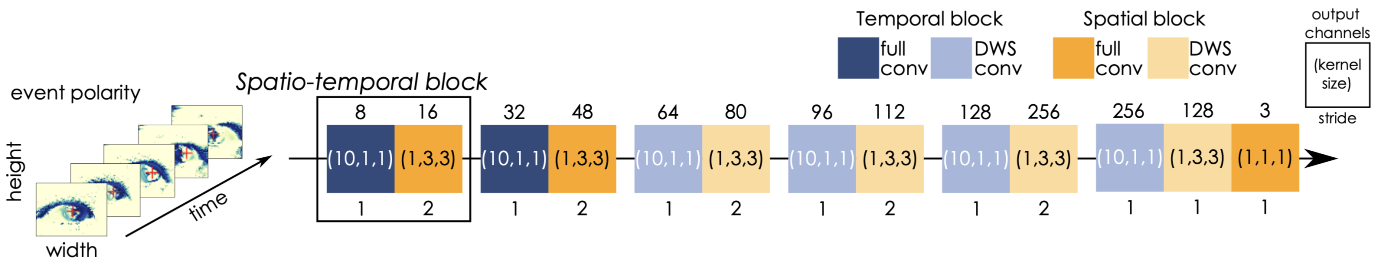

Fig. 1. Architecture of the TENNs-PLEIADES model used for gesture recognition on DVS128 data. The network is composed of five spatiotemporal blocks, each consisting of a temporal convolution (left) followed by a spatial convolution (right). Temporal convolutions (kernel size 10×1×1) process the sequence of 10 ms event frames along the time axis, while spatial convolutions (kernel size 1×3×3) extract local features across the spatial plane. Dark blocks represent full convolutions, and lighter blocks indicate depthwise-separable (DWS) convolutions, used to reduce computational overhead. The network progressively downsamples the spatial resolution while increasing feature depth, ultimately projecting to gesture class logits.

goes a 3×3 convolution across spatial dimensions to integrate local pixel information. These spatial convolutions are also optionally depthwise-separable, reducing MACs and enhancing efficiency. Most blocks apply a stride of 2 to downsample the input, progressively reducing the original 128×128 resolution to a 4×4 feature map by the final block. This is followed by Batch Normalization—appropriate here since spatial operations are not temporally causal—and another ReLU activation. After the fifth block, the resulting feature volume (4×4×256) undergoes global average pooling to yield a 256-dimensional vector for each frame. A lightweight two-layer MLP, implemented using two 1×1 temporal convolutions with a ReLU in between, projects this vector to gesture logits. The network produces predictions at 10 ms intervals, and thanks to the deep temporal receptive field built up across layers, it captures long-range motion without needing any recurrent modules like LSTMs or ConvLSTMs.

One of the core design principles of PLEIADES is its commitment to causality and low latency. Since each of the five spatiotemporal blocks contains a temporal kernel of size 10, the model's full receptive field requires a warm-up latency of 440 ms (5 × (10–1) steps) to ensure all temporal features are valid. However, through techniques like zero-padding or frame-masking, the model can begin producing outputs after only 100–200 ms, with minimal loss in accuracy. Post-processing options like majority filtering can also be used to further refine predictions. Despite its efficiency-focused architecture, PLEIADES achieves 99.59% raw test accuracy on the DVS128 dataset using only 192k parameters, outperforming many larger models while remaining exceptionally suitable for real-time deployment on edge hardware.

B. *S-TLLR model architecture*

STDP-inspired Temporal Local Learning Rule (S-TLLR) is a biologically inspired method, designed to train spiking neural networks (SNNs) in an efficient way, while maintaining important temporal dynamics. It is particularly suitable for low-power and online learning scenarios, due to its low memory requirements and its ability to operate without storing full sequences of past activity. It draws inspiration from Spike-Timing Dependent Plasticity (STDP), a biologically-plausible learning rule in which the strength of a synapse is adjusted depending on the relative timing of spikes between connected neurons. However, unlike traditional STDP, S-TLLR incorporates supervision by modulating weight updates with an error signal, and it is temporally local, so it does not rely on accessing future time steps or storing full histories of neural activity. Because it needs only two short traces per neuron, its memory cost is O(n) - suitable for low-power, real-time hardware.

At each time step during training, S-TLLR computes weights updates based on the presynaptic activity, representing the recent input of neuron, the postsynaptic activity, reflecting how close the neuron is to firing, and a learning signal δ, that reflects the network's output error.

The first two components are used to compute a short-term eligibility trace ($e_{ij}[t]$) - a local, synapse-specific signal that captures how strongly a synapse has been involved in recent activity, which reflects both causal and non-causal spike timing relationship, allowing the model to learn from both cause-effect dynamics and synchrony (Figure 2).

The learning signal plays the role of task-specific supervision, informing each neuron wether its recent activity contributed to a correct or incorrect output, and it can be computed with two different approaches:

- Backpropagation throught Layers (BP): In this mode, the network computes the error of the output layer by comparing its output activity to the target label. The error is then back-propagated from layer to layer, using a standard chain rule. However, unlike traditional BPTT, there is no need to store the entire history of network activity. Each neuron receives a layer-specific, time-local learning signal that approximates how much its current activity affects the final output.
- Direct Feedback Alignment (DFA): In this biologically plausible alternative, the backpropagation is ignored altogether. Instead of propagating error signals layer by layer, DFA sends the error from the output layer directly to each hidden layer through a fixed, randomly initialized matrix, so the signal is not dependent on the forward weights and does not require any gradient computation in the intermediate layers.

Once the eligibility trace and the learning signal are computed at a given time step, they are combined to produce the final weight update, following a three-factor rule ($\Delta w_{ij} \propto \delta_i[t]\, e_{ij}[t]$), in which the synaptic change is proportional to the product of the eligibility trace and the learning signal.

S-TLLR integrates easily with various spiking network architectures. For vision-based event datasets, including the DVS Gesture, the architecture is a spiking version of a standard convolution neural network, adapted from the traditional VGG-9 architecture. The events streams are accumulated into temporal bins, typically over a 1.5-second window. Each bin represents a short time interval, and multiple bins form a sequence. Events are spatially mapped into frames of 32x32 pixels, with two channels representing positive and negative polarities, which results in a 3D input tensor of shape: [Time x Channels x Height x Width], such as [20 x 2 x 32 x 32]. The spiking VGG-9 architecture used for this task consist of 8 convolutional layers with 3x3 kernels, arranged in blocks with progressively increasing feature maps (64 to 512 channels). Each block is followed by 2×2 average pooling, which reduces spatial resolution while maintaining temporal structure. All layers use Leaky Integrate-and-Fire (LIF) neurons, which integrate spikes over time and reset after firing.

The final convolutional output is flattened and passed to a fully connected LIF output layer, producing one output unit per gesture class. Outputs are aggregated across time, typically by spike count or membrane potential integration at the last few steps.

This architecture enables hierarchical spatial feature extraction while capturing temporal dynamics through frame-by-frame spiking activity, making it well suited for real-time gesture recognition.

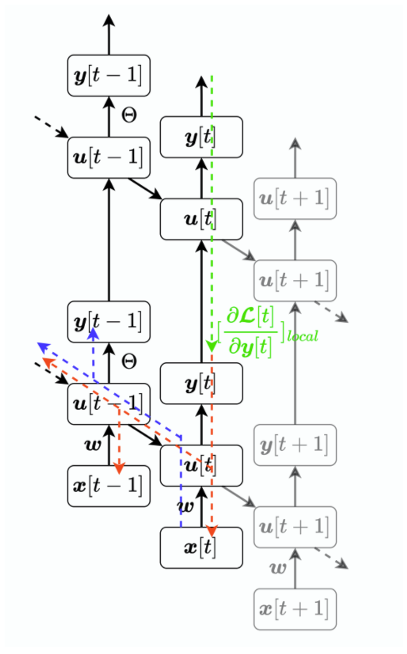

Fig. 2. Temporally local weight update in S-TLLR. Causal and non-causal spike timing contributions (red and blue dashed arrows) form the eligibility trace, which is modulated by a local learning signal (green) at the same time step. Future states (grey) are not required.

IV. Experiments and Results

While both PLEIADES and S-TLLR are designed for gesture recognition using event-based data, their training philosophies differ fundamentally. PLEIADES is trained with gradients applied at every timestep, encouraging the model to make confident predictions as early as possible. This design aligns naturally with its use of causal temporal convolutions and ReLU activations, which allow each frame to influence the output immediately. In contrast, S-TLLR applies its loss only at the final timestep of the sequence, training the network to focus on end-of-sequence confidence rather than early responsiveness. Additionally, while CNN layers in PLEIADES react instantly with each input, spiking neurons in S-TLLR require the gradual accumulation of membrane potential and spike-based integration, inherently slowing down the decision-making process. Moreover, PLEIADES uses zero-padding around its 10-frame causal kernels, allowing it to start producing valid outputs early. S-TLLR, however, is trained on full 1.5s sequences and expects long-term temporal context—cutting.

Due to these differences, direct comparison required retraining S-TLLR under a setup that allows fine-grained temporal evaluation. The original S-TLLR release only provided epoch-level .npy logs and did not include the trained model weights or time-resolved predictions. To enable latency analysis similar to PLEIADES, we retrained the S-TLLR model from scratch and saved the best-performing checkpoint. With this retrained model, we processed each gesture sequence across 20 discrete timesteps (each 75ms) and evaluated the model's predictions at each increment from 75ms up to the full 1.5s duration. This approach allowed us to fairly measure accuracy as a function of latency for both models, revealing their distinct temporal response profiles under the same experimental conditions.

In terms of overall accuracy, PLEIADES achieves a perfect 100% final accuracy, while S-TLLR reaches a slightly lower yet still impressive 97.92%. This result is expected considering the fundamental differences in training philosophy. PLEIADES processes event frames at 10ms intervals and is explicitly trained to produce per-frame predictions with full causal context, allowing it to make highly confident decisions by the end of the gesture sequence. In contrast, S-TLLR is trained to emit a single final prediction after observing the entire 1.5s input, meaning it optimizes for end-of-sequence confidence rather than per-frame responsiveness. As a result, while S-TLLR doesn't quite reach PLEIADES' ceiling, it still performs remarkably well despite being built on a spiking architecture that inherently involves sparse, non-linear activation and membrane integration dynamics.

When we examine the latency vs accuracy curve in Figure 3, PLEIADES rapidly ramps up in accuracy after around 100–150ms and surpasses S-TLLR after about 200–300ms, it initially lags behind. Interestingly, S-TLLR actually shows slightly higher accuracy at the very first few latency points (e.g., less than 100ms), despite being trained only to predict at the end of the sequence. This counterintuitive early advantage can be attributed to the nature of spiking neurons, which can sometimes emit early spikes in response to salient stimulus patterns without needing full temporal context. These early spikes can accidentally align with the correct class, offering a head start in low-latency settings. Meanwhile, PLEIADES requires its 10-frame causal temporal kernels to ”warm up” — meaning its initial outputs (before 100ms) are influenced by zero-padding and lack meaningful temporal evidence. Once the receptive field is populated, however, PLEIADES quickly overtakes and maintains superior accuracy, reinforcing its design for real-time, low-latency inference.

V. Conclusion

In this work, we compared two fundamentally different approaches to event-based gesture recognition: the PLEIADES model, a spatiotemporal convolutional network using parameterized polynomial kernels, and S-TLLR, a spiking neural network trained with a biologically inspired local learning rule.

PLEIADES achieves high accuracy with low-latency through efficient spatiotemporal convolutions and causal temporal kernels, making it ideal for fast, frame-wise predictions, and begins producing accurate predictions shortly after its temporal receptive field is populated (around 100–150 ms), eventually surpassing S-TLLR in accuracy.

In contrast, S-TLLR offers a lightweight, memory-efficient alternative using local, online learning without backpropagation through time. While it requires longer temporal context to reach peak performance, it displays competitive early predictions—even outperforming PLEIADES at very short latencies despite being trained only for end-of-sequence classification, likely due to the event-driven dynamics of spiking neurons.

Overall, PLEIADES offers superior accuracy and responsiveness in time-critical applications, while S-TLLR provides a competitive, low-memory alternative, which aligns well with real-time applications.

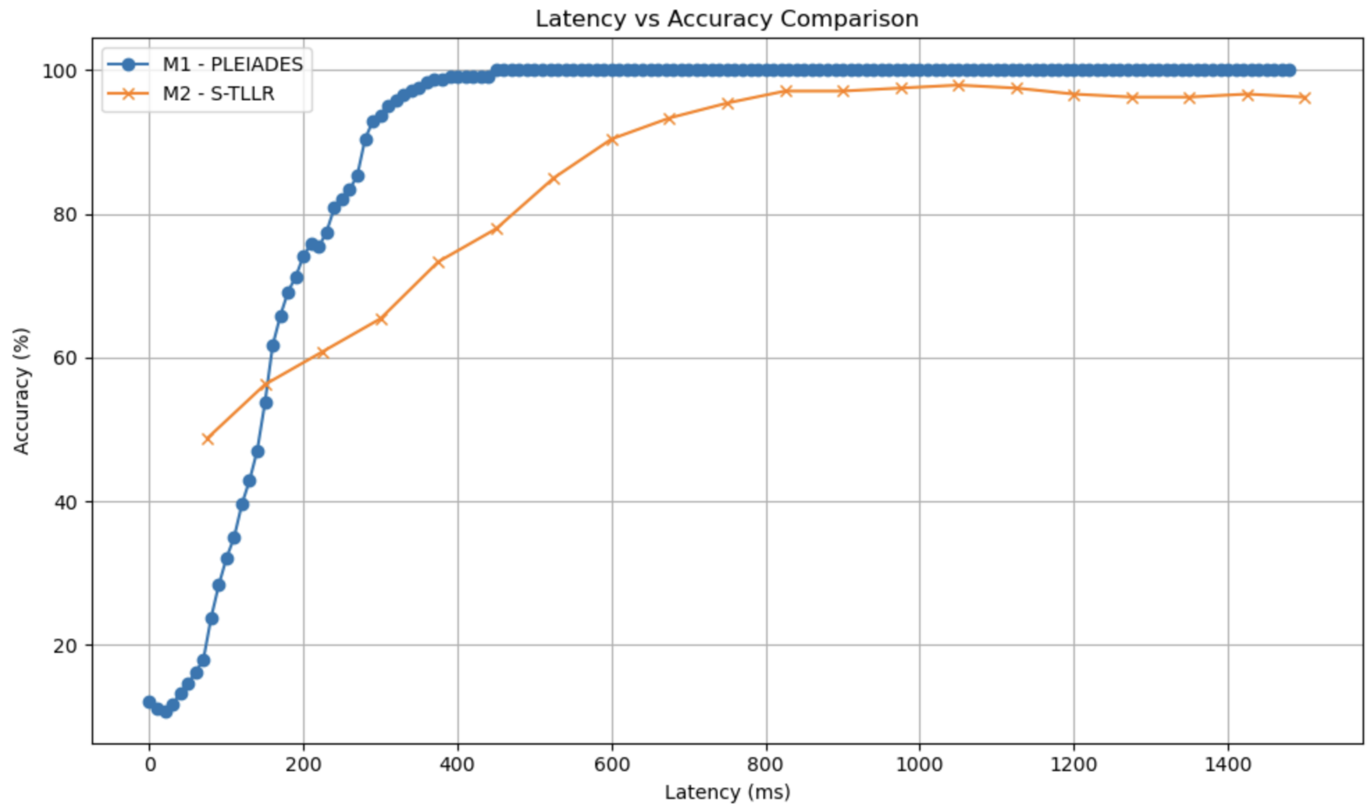

Fig. 3. Latency vs. accuracy comparison between PLEIADES (M1) and S-TLLR (M2) on the DVS128 gesture dataset.

The results highlight a trade-off between fast, per-frame prediction and efficient, biologically inspired sequence-level learning, with each model excelling under different deployment constraints.

VI. Author credits

Ginevra Bozza: Literature review (SNNs, Event-based data), Methodology (S-TLLR), Conclusion. **Saijal Singhal**: Introduction, Literature Review (Temporal Kernels), Methodology (TENNs-PLEIADES), Experiments and Results

References

[1] S. R. Sabbella, S. Kaszuba, F. Leotta, P. Serrarens, and D. Nardi, "Evaluating gesture recognition in virtual reality," *arXiv preprint arXiv:2401.04545*, 2024.

[2] A. S. M. Miah, M. A. M. Hasan, Y. Tomioka, and J. Shin, "Hand gesture recognition for multi-culture sign language using graph and general deep learning network," *IEEE Open Journal of the Computer Society*, 2024.

[3] Q. Hu, "Enhancing american sign language communication with virtual reality: A gesture recognition application on oculus quest 2," Ph.D. dissertation, Carleton University, 2024.

[4] Y. R. Pei and O. Coenen, "Tenns-pleiades: Building temporal kernels with orthogonal polynomials," *arXiv preprint arXiv:2405.12179*, 2024.

[5] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7243–7252.

[6] Y. Li, Q. Miao, K. Tian, Y. Fan, X. Xu, Z. Ma, and J. Song, "Large-scale gesture recognition with a fusion of rgb-d data based on optical flow and the c3d model," *Pattern recognition letters*, vol. 119, pp. 187–194, 2019.

[7] C.-C. Hsieh, D.-H. Liou, and D. Lee, "A real time hand gesture recognition system using motion history image," in *2010 2nd international conference on signal processing systems*, vol. 2. IEEE, 2010, pp. V2–394.

[8] D. Ojha and R. G. Rajan, "Histogram based human computer interaction for gesture recognition," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2019, pp. 263–266.

[9] F. Rahman, J. Samuel, and N. McFarlane, "Comparison of svm and random forest for multimodal hand gesture recognition," in *SoutheastCon 2025*. IEEE, 2025, pp. 924–929.

[10] A. K. Singh, "Hidden markov model for gesture recognition," in *Challenges and Applications for Hand Gesture Recognition*. IGI Global Scientific Publishing, 2022, pp. 109–123.

[11] H. PA, I. A. TP *et al.*, "Hand gesture recognition for indian sign language: A cnn-gru-mlp approach." *International Journal of Intelligent Engineering & Systems*, vol. 18, no. 4, 2025.

[12] F. Al Farid, N. Hashim, J. B. Abdullah, M. R. Bhuiyan,

M. Kairanbay, Z. Yusoff, H. A. Karim, S. Mansor, M. T. Sarker, and G. Ramasamy, "Single shot detector cnn and deep dilated masks for vision-based hand gesture recognition from video sequences," *IEEE Access*, vol. 12, pp. 28 564–28 574, 2024.

[13] D. Sarma, V. Kavyasree, and M. Bhuyan, "Two-stream fusion model using 3d-cnn and 2d-cnn via video-frames and optical flow motion templates for hand gesture recognition," *Innovations in Systems and Software Engineering*, vol. 21, no. 1, pp. 39–52, 2025.

[14] A. Toro-Ossaba, J. Jaramillo-Tigreros, J. C. Tejada, A. Peña, A. López-González, and R. A. Castanho, "Lstm recurrent neural network for hand gesture recognition using emg signals," *Applied Sciences*, vol. 12, no. 19, p. 9700, 2022.

[15] M. Montazerin, E. Rahimian, F. Naderkhani, S. F. Atashzar, S. Yanushkevich, and A. Mohammadi, "Transformer-based hand gesture recognition from instantaneous to fused neural decomposition of high-density emg signals," *Scientific reports*, vol. 13, no. 1, p. 11000, 2023.

[16] A. S. M. Miah, M. A. M. Hasan, and J. Shin, "Dynamic hand gesture recognition using multi-branch attention based graph and general deep learning model," *IEEE Access*, vol. 11, pp. 4703–4716, 2023.

[17] M. Wu, "Gesture recognition based on deep learning: A review." *EAI Endorsed Transactions on e-Learning*, vol. 10, 2024.

[18] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[19] Z. Wang, J. Yao, M. Xu, M. Jiang, and J. Su, "Transformer-based network with temporal depthwise convolutions for semg recognition," *Pattern Recognition*, vol. 145, p. 109967, 2024.

[20] G. Chen, Z. Dong, J. Wang, and J. Hu, "A resource-efficient partial 3d convolution for gesture recognition," *Journal of Real-Time Image Processing*, vol. 21, no. 4, p. 132, 2024.

[21] G. Rutishauser, M. Scherer, T. Fischer, and L. Benini, "7 μj/inference end-to-end gesture recognition from dynamic vision sensor data using ternarized hybrid convolutional neural networks," *Future Generation Computer Systems*, vol. 149, pp. 717–731, 2023.

[22] M. P. E. Apolinario and K. Roy, "S-tllr: Stdp-inspired temporal local learning rule for spiking neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2306.15220

[23] C. Yu, Z. Gu, D. Li, G. Wang, A. Wang, and E. Li, "Stsc-snn: Spatio-temporal synaptic connection with temporal convolution and attention for spiking neural networks," *Frontiers in Neuroscience*, vol. 16, Dec. 2022. [Online]. Available: http://dx.doi.org/10.3389/fnins.2022.1079357

[24] Z. Zhao, Y. Wang, Q. Zou, T. Xu, F. Tao, J. Zhang, X. Wang, C. J. R. Shi, J. Luo, and Y. Xie, "The spike gating flow: A hierarchical structure based spiking neural network for online gesture recognition," 2022. [Online]. Available: https://arxiv.org/abs/2206.01910

[25] M. Yao, G. Zhao, H. Zhang, Y. Hu, L. Deng, Y. Tian, B. Xu, and G. Li, "Attention spiking neural networks," 2022. [Online]. Available: https://arxiv.org/abs/2209.13929

[26] S. Yang, S. Lu, S. Wang, M. H. Er, Z. Zheng, and A. C. Kot, "Temporal-guided spiking neural networks for event-based human action recognition," 2025. [Online]. Available: https://arxiv.org/abs/2503.17132

[27] B. Chakravarthi, A. A. Verma, K. Daniilidis, C. Fermuller, and Y. Yang, "Recent event camera innovations: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2408.13627

[28] A. K. Kosta and K. Roy, "Adaptive-spikenet: Event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics," 2023. [Online]. Available: https://arxiv.org/abs/2209.11741