

MASTER'S THESIS ARTIFICIAL INTELLIGENCE

# Event-based Near-eye Gaze Tracking using a Spiking Neural Network

STIJN GROENEN

8 January 2025

*Supervisor:*

Marzieh Hassanshahi Varposhti

*Supervisor:*

Mahyar Shamsavari

*Second reader:*

Qinyu Chen

Radboud University



# GazeSCRNN: Event-based Near-eye Gaze Tracking using a Spiking Neural Network

Stijn Groenen  
Radboud University  
Nijmegen, The Netherlands  
ORCID: 0009-0003-7042-2954

Marzieh Hassanshahi Varposhti  
Donders Institute for Brain,  
Cognition, and Behaviour  
Nijmegen, The Netherlands  
ORCID: 0009-0005-7958-2051

Mahyar Shahsavari  
Donders Institute for Brain,  
Cognition, and Behaviour  
Nijmegen, The Netherlands  
ORCID: 0000-0001-7703-6835

**Abstract**—This work introduces GazeSCRNN, a novel spiking convolutional recurrent neural network designed for event-based near-eye gaze tracking. Leveraging the high temporal resolution, energy efficiency and compatibility of Dynamic Vision Sensor (DVS) cameras with event-based systems, GazeSCRNN uses a spiking neural network (SNN) to address the limitations of traditional gaze-tracking systems in capturing dynamic movements. The proposed model processes event streams from DVS cameras using Adaptive Leaky-Integrate-and-Fire (ALIF) neurons and a hybrid architecture optimized for spatio-temporal data. Extensive evaluations on the EV-Eye dataset demonstrate the model’s accuracy in predicting gaze vectors. In addition, we conducted ablation studies to reveal the importance of the ALIF neurons, dynamic event framing, and advanced training techniques, such as Forward-Propagation-Through-Time, in enhancing overall system performance. The most accurate model achieved a Mean Angle Error (MAE) of  $6.034^\circ$  and a Mean Pupil Error (MPE) of 2.094 mm. Consequently, this work is pioneering in demonstrating the feasibility of using SNNs for event-based gaze tracking, while shedding light on critical challenges and opportunities for further improvement.

**Index Terms**—Gaze tracking, Spiking neural network, Event-based vision, Neuromorphic computing

## I. INTRODUCTION

Eye and gaze tracking play a crucial role in various fields, including human-computer interaction, robotics, accessibility for users with motor disabilities, virtual reality (VR), and mixed reality (MR). These technologies leverage eye movements to create more intuitive and immersive experiences. However, traditional eye-tracking systems often face limitations in speed and energy efficiency [1]–[3].

To address these limitations, Dynamic Vision Sensor (DVS) cameras have emerged as a promising alternative. These event-based cameras offer advantages over conventional frame-based cameras, such as high temporal resolution, low latency, and energy efficiency. DVS cameras achieve this by registering and emitting events only when changes in pixel brightness occur, unlike traditional cameras that capture images at a fixed frame rate regardless of scene changes [4], [5].

Spiking Neural Networks (SNNs) are uniquely suited to process the event-driven data generated by DVS cameras. SNNs operate using sparse computations where information is encoded and transmitted as discrete spikes, mirroring the biological neural processes in the human brain [6]. This results

in significant gains in processing speed and energy efficiency, making SNNs well-suited for handling the spatio-temporal data from DVS cameras [7].

This work presents a pioneering investigation into the potential of DVS cameras and SNNs for fast and efficient near-eye gaze tracking. The research focuses on a distinct SNN architecture: a spiking convolutional recurrent neural network, which is designed to predict the user’s gaze vector.

The study uses the EV-Eye dataset [8], which comprises event streams recorded by DVS cameras aimed at participants’ eyes during several gaze-related tasks. During data preprocessing, the events are aggregated into frames and aligned with the ground truth gaze references. Model performance is evaluated based on two primary metrics: Mean Pupil Error (MPE), which measures the distance between the predicted origin point and the actual pupil coordinates, and Mean Angle Error (MAE), which quantifies the angular difference between the predicted and ground truth gaze vectors.

The training process utilizes backpropagation and stochastic gradient descent, incorporating the surrogate gradient method to manage the non-differentiable nature of spikes [9]. Given the time-dependent characteristics of the models, either truncated Backpropagation-Through-Time (BPTT) or Forward-Propagation-Through-Time (FPTT) is employed, allowing for efficient training while managing memory constraints [10], [11].

The source code for our models and experiments are available at <https://github.com/StijnGroenen/GazeSCRNN>.

## II. RELATED WORK

### A. Gaze tracking

Gaze-tracking technologies have been categorized by Liu et al. [2] into two primary approaches: model-based and appearance-based techniques. These approaches address the gaze tracking task through fundamentally different strategies, each with unique benefits and limitations based on the application environment, hardware requirements, and tolerance for head motion. Additionally, gaze tracking systems can be divided into remote setups, which rely on an external camera, and near-eye or head-mounted setups, for which the camera moves with the user’s head, limiting the increased difficulty of handling head movements [2].

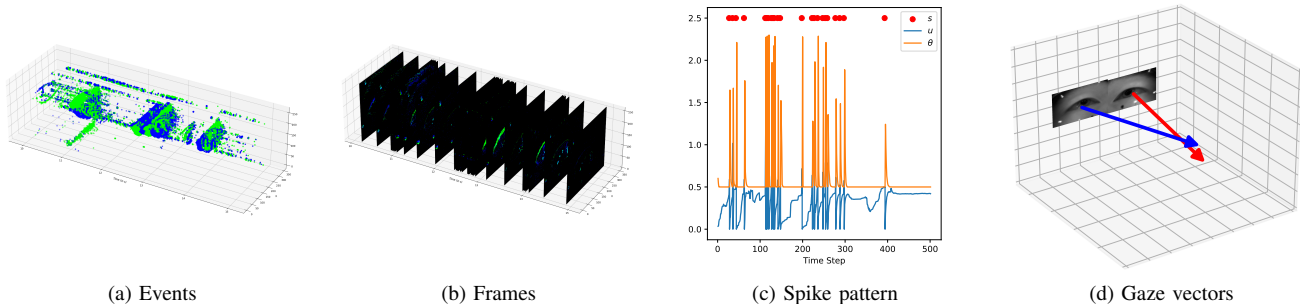


Fig. 1. Overview of the event-based gaze tracking pipeline. 1a shows input events captured by a Dynamic Vision Sensor (DVS) camera. 1b shows the events aggregated into frames during data preprocessing. 1c shows frames processed by the spiking neural network, producing sequences of complex spike patterns. 1d shows output gaze vectors, indicating the user’s point of focus.

Model-based gaze tracking methods use geometric models of the eye to estimate gaze direction, relying on specific eye features such as the pupil and iris location, eye corner points, and glints from (infrared) illumination. Model-based methods can be categorized into 2D mapping-based and 3D model-based approaches [2]. For 2D mapping-based methods, a mapping function is constructed between certain 2D eye features, such as pupil or iris position, and gaze points on a screen. This requires personal calibration, making it sensitive to head motion effects, since the mapping is based on a fixed head position relative to the screen. Alternatively, 3D model-based methods leverage geometric relationships and the 3D structure of the eyeball to reconstruct the line-of-sight.

In contrast to model-based methods, appearance-based gaze tracking techniques rely on direct image analysis rather than explicit geometric models of the eye. These methods use the visual appearance of the eye, such as the eye shape, pupil, and iris patterns to predict gaze direction, which enables a more flexible approach suitable for a wide range of environments, often without a strict requirement for personal calibration or infrared illumination [12]. Neural networks, particularly convolutional neural networks (CNNs), have shown promising results in appearance-based gaze tracking. CNNs use convolutional layers to automatically extract hierarchical features from eye images, making them robust to variations in lighting and head poses. However, they require extensive computational resources and a large labelled dataset for effective training [12]. Furthermore, challenges remain in achieving real-time performance on lightweight devices. In this paper, we train a spiking neural network (SNN) for event-based near-eye gaze tracking, that can theoretically be run on efficient neuromorphic hardware. Furthermore, our solution does not use explicit user calibration. However, it is possible to apply personal calibration through an online learning approach, allowing the SNN to become more optimized over time for a specific user.

### B. Event-based Vision and Spiking Neural Networks for Gaze Tracking

Dynamic Vision Sensor (DVS) cameras and Spiking Neural Networks (SNNs) represent a promising combination. DVS cameras emit events only when changes in pixel brightness occur, which provides high temporal resolution, low latency, and energy efficiency [4], [5]. These characteristics align well with the sparse and event-driven processing model of SNNs, which encode information as discrete spikes that mimic biological neural activity. This can lead to significant improvements in computational costs and energy efficiency compared to traditional artificial neural networks [7].

Despite this natural synergy, much of the prior work with DVS cameras for eye and gaze tracking has relied on conventional processing techniques. For example, Angelopoulos et al. [13] combined regular frames and DVS events to dynamically update a pupil model at high speeds, and then estimate the corresponding gaze vectors using a user-calibrated polynomial regression. Ryan et al. [14] reconstructed frames from DVS events, and applied a convolutional recurrent neural network to predict face and eye positions. Similarly, Chen et al. [15] proposed a Change-Based ConvLSTM network for efficient pupil tracking.

Several works highlight the potential of integrating SNNs and DVS cameras for eye tracking systems. Jiang et al. [16] demonstrated an SNN-based near-eye eye-tracking model that leverages the asynchronous temporal information from event cameras, achieving enhanced stability and accuracy with low computational costs. Similarly, Bonazzi et al. [17] proposed an SNN for pupil tracking, implemented on neuromorphic hardware to exploit the low-latency and energy efficiency of the event-based paradigm. However, prior research into SNNs was limited to eye tracking, rather than gaze tracking.

This study builds upon all these efforts by proposing a novel spiking convolutional recurrent neural network architecture optimized for event-based near-eye gaze tracking, leveraging the temporal precision and energy efficiency of DVS cameras, as well as the sparse, and energy-efficient computations of SNNs, with the capability to be implemented on neuromorphic and edge devices for real-time gaze tracking and online learning.

### III. METHODS

#### A. EV-Eye Dataset

For training and evaluating our model, we use the event-based EV-Eye dataset introduced by Zhao et al. (2023) [8]. The dataset features asynchronous event streams collected using a pair of DAVIS346 Dynamic Vision Sensor (DVS) cameras with a resolution of  $346 \times 260$  [18] (see Figure 1a), one dedicated to each eye, positioned to capture participants' eye movements while they perform specific gaze tasks.

The EV-Eye dataset covers three gaze tasks that simulate common eye movement patterns [8]:

- 1) Smooth Pursuit: Participants follow a moving stimulus across a screen in a smooth, predictable trajectory.
- 2) Random Saccade: Participants search for a stimulus that appears at random locations on the screen.
- 3) Fixation: Participants maintain their gaze on a stationary stimulus for an extended period, achieved by keeping the stimulus in one position during the random saccade task.

Data was gathered from 48 participants, each completing two sessions for both the smooth pursuit and random saccade tasks, resulting in a total of 192 event streams. Alongside the event data, the dataset provides gaze references in the form of vectors recorded by Tobii Pro Glasses 3 at 100Hz [19], as depicted in Figure 1d. Zhao et al. [8] argue that gaze vectors provide a more accurate reference compared to screen coordinates, noting that eye movements often diverge from the exact location of the stimulus on a screen, particularly during the search phase of the random saccade task.

1) *Data Preprocessing*: Prior to training our model, we preprocess the event streams from the DVS cameras by aggregating them into frames (see Figure 1b). This aggregation is either performed using fixed time windows or by using a fixed number of events per frame (see Section IV-B).

For most of our models, the frames are downsampled by a factor of two in order to improve efficiency, resulting in final dimensions of  $173 \times 130 \times 2$ . The two channels represent the positive and negative polarity events, respectively.

The gaze references provided in the EV-Eye dataset are utilized to compute the loss function. These references consist of a 3D origin point and two spherical coordinates representing the gaze direction (omitting the radius). The spherical coordinates are normalized so that  $0^\circ$  corresponds to the user looking straight ahead. The gaze references are aligned with the aggregated frames as close as possible using their timestamps, and are linearly interpolated between true references when the timestamps do not line up exactly.

However, since the gaze references in the EV-Eye dataset are predictions from another gaze tracker, they may not perfectly reflect the user's true gaze. Furthermore, we observed prolonged periods where the device failed to provide valid gaze vectors (max. 8.5s), leading to interpolation between widely separated ground truth values. These interpolated values can be unreliable. To address this, we optionally mask these interpolated references for some models, if the time window between the ground truth references exceeds a pre-defined

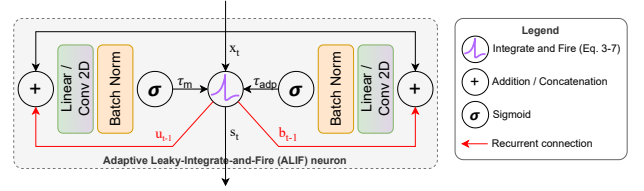


Fig. 2. Adaptive Leaky-Integrate-and-Fire Neuron with Liquid Time Constants

threshold, excluding them from loss and metric computations (see Section IV-E).

To manage memory constraints and enable fixed-size training batches, we divide the event streams into sequences of 1000 frames. These sequences are organized into training (70%), validation (15%), and testing (15%) sets by randomly selecting sequences for validation and testing using a fixed seed for reproducibility.

In an attempt to improve generalizability and robustness against noise, we apply optional data augmentation to the aggregated frames (see Section IV-D). Augmentations include random rotations (up to  $2^\circ$ ), translations (up to 1%), rescaling (up to 1%), and shearing (up to  $2^\circ$ ). Additionally, random noise events can be injected based on a configurable probability hyperparameter.

#### B. Spiking Neurons

There are various types of spiking neurons. The most commonly used spiking neuron model is the Leaky-Integrate-and-Fire (LIF) neuron. As the name suggests, LIF-neurons integrate the incoming spikes to update a leaky membrane potential which decays over time, and it fires (and resets) when the membrane potential reaches a threshold. Instead of regular Leaky-Integrate-and-Fire (LIF) neurons, our models use two distinct types of spiking neuron models: Parametric Leaky-Integrate-and-Fire (PLIF) neurons and Adaptive Leaky-Integrate-and-Fire (ALIF) neurons. ALIF neurons are adaptive, meaning their spiking behaviour responds dynamically to the input (see Figure 1c). We base our implementation on the models by Yin et al. [11], [20], which incorporate Liquid Time Constants (LTCs) that enable dynamic adjustment of both membrane potential decay and the firing threshold.

In these adaptive neurons, the membrane potential decay constant  $\tau_m$  and the adaptation decay constant  $\tau_{adp}$  are computed at each time step based on the input and neuron state. This gives ALIF neurons liquid time constants that vary with the network's input, which allows them to track both short- and longer-term dependencies within the sequence. The result is a neuron model that captures richer temporal dynamics than standard spiking neurons [20].

PLIF neurons, in contrast, use fixed parameters for  $\tau_m$  and  $\tau_{adp}$ , which are optimized during training but do not vary with input at each step.

Equations 1 through 7 describe the computations governing both types of neurons.

$$\tau_m = \begin{cases} \sigma(\text{Conv2D}(x_t + u_{t-1})) & \text{if } x_t \text{ is 2D} \\ \sigma(\text{Linear}(x_t \parallel u_{t-1})) & \text{if } x_t \text{ is 1D} \end{cases} \quad (1)$$

$$\tau_{adp} = \begin{cases} \sigma(\text{Conv2D}(x_t + b_{t-1})) & \text{if } x_t \text{ is 2D} \\ \sigma(\text{Linear}(x_t \parallel b_{t-1})) & \text{if } x_t \text{ is 1D} \end{cases} \quad (2)$$

$$u_t = u_{t-1} + \tau_m(x_t - u_{t-1}) \quad (3)$$

$$b_t = \tau_{adp}b_{t-1} + (1 - \tau_{adp})s_{t-1} \quad (4)$$

$$\theta_t = b_0 + \beta b_t \quad (5)$$

$$s_t = \Theta(u_t - \theta_t) \quad (6)$$

$$u_t = u_t(1 - s_t) + u_r s_t \quad (7)$$

The parameter  $\tau_m$  in Equation 1 is the membrane time constant for the neuron, controlling the rate at which the neuron’s membrane potential  $u_t$  decays over time. For ALIF neurons, the time constant can vary adaptively at each time step and is calculated as a function of the current input  $x_t$  and the previous membrane potential  $u_{t-1}$ . Specifically, if the input  $x_t$  is two-dimensional (excluding channels; such as spatially structured data, like frames), a sigmoid function  $\sigma$  and convolutional layer is used to compute the time constants, whereas for one-dimensional inputs, a fully-connected layer is used. For PLIF neurons,  $\tau_m$  is not adaptive, but is a trained parameter.

The parameter  $\tau_{adp}$  in Equation 2 is the adaptation time constant, which controls the rate at which the adaptation variable  $b_t$  decays. Like  $\tau_m$ , for ALIF neurons, this time constant is also input-dependent, computed based on  $x_t$  and the adaptation variable from the previous time step  $b_{t-1}$ .

The parameter  $u_t$  in Equation 3 is the membrane potential of the neuron at time  $t$ , representing the internal state that integrates input signals. It is updated based on the previous membrane potential  $u_{t-1}$ , the current input  $x_t$ , and the membrane time constant  $\tau_m$ , which scales the influence of the current input on the potential decay rate.

The variable  $b_t$  in Equation 4 is an adaptation variable that represents the dynamic adjustment of the neuron’s firing threshold. It is updated based on the previous adaptation state  $b_{t-1}$ , the adaptation time constant  $\tau_{adp}$ , and the previous spike output  $s_{t-1}$ . This adaptation variable increases with spiking activity, raising the threshold temporarily to reduce the firing rate.

The firing threshold  $\theta_t$  in Equation 5 at time  $t$  is defined as the baseline threshold  $b_0$  plus a scaling factor  $\beta$  applied to the adaptation variable  $b_t$ . This adaptive threshold adjusts dynamically to control the neuron’s sensitivity to inputs, preventing excessive spiking. Like in [20],  $\beta$  is a hyperparameter that defaults to 1.8.

The binary variable  $s_t$  in Equation 6 represents the spike output of the neuron at time  $t$ . It is computed by the Heaviside step function  $\Theta$ , which outputs 1 (a spike) if the membrane potential  $u_t$  exceeds the threshold  $\theta_t$ , and 0 otherwise.

Equation 7 updates the membrane potential  $u_t$  after a spike. When  $s_t = 1$ , indicating a spike,  $u_t$  is reset to the resting potential  $u_r$  (a hyperparameter that defaults to 0), otherwise,  $u_t$  remains unchanged. This mechanism ensures that the neuron’s membrane potential is reset after each spike. In our implementation, it is also possible to disable the reset mechanism, for example for the final layer, where the membrane potential of the neurons can be used as the output of the model.

Figure 2 depicts the internal structure of an ALIF neuron layer with Liquid Time Constants.

### C. GazeSCRNN: Spiking Convolutional Recurrent Neural Network Architecture

In this work, we propose the GazeSCRNN architecture: a spiking convolutional recurrent neural network designed for event-based gaze tracking. Since the event data contains both spatial and temporal dimensions, this architecture employs a combination of convolutional and recurrent layers to process the event data and predict the user’s gaze vector. Adaptive Leaky-Integrate-and-Fire (ALIF) neurons, as described in Section III-B, are used in this model to allow the spiking behaviour of neurons to adapt to the input data and be optimized during training [20], adding complexity to the otherwise relatively simple architecture.

Figure 3 shows a diagram of the GazeSCRNN architecture. The network accepts a 2-channel input tensor of dimensions  $2 \times 130 \times 173$ , representing the downsized aggregated frames derived from the events captured by the DVS camera. Spatial features are extracted through a series of four convolutional blocks, each comprising a convolutional layer, batch normalization, max-pooling (for the first two blocks), and ALIF neurons. In addition to this model, a version trained on full-sized input frames of  $2 \times 260 \times 346$  includes an additional convolutional block to extract spatial features at the larger scale, followed by a max-pooling layer to downscale the spatial dimensions by a factor of two.

The convolutional blocks progressively reduce the spatial dimensions while increasing the number of filters to extract progressively richer features, starting with 32 filters in the first block and scaling up to 128 filters in the final block. Batch normalization is applied after each convolutional layer to enhance training stability, and max-pooling layers reduce the spatial dimensions for efficient processing. ALIF neuron layers integrate incoming values from the convolutional layers into their membrane potentials and fire when the spike threshold is exceeded, introducing sparsity and efficiency to the computation. Dropout regularization is applied after the last convolutional block to reduce overfitting.

Temporal dependencies in the event stream are modelled by four recurrent blocks, each consisting of a fully connected layer, batch normalization, and ALIF neurons. These recurrent blocks integrate temporal information by concatenating their input features with the spiking outputs from the previous time step, enabling the model to retain and refine temporal representations across multiple time steps.

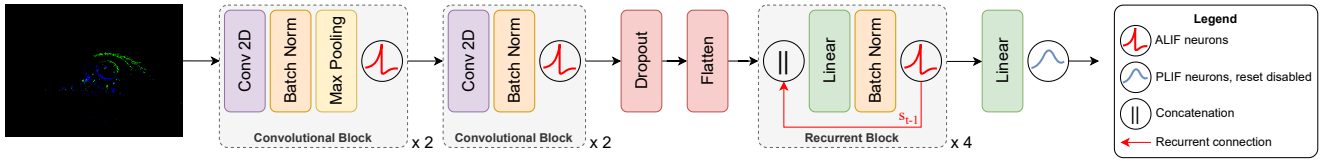


Fig. 3. GazeSCRNN, a Spiking Convolutional Recurrent Neural Network architecture for event-based gaze tracking

The final gaze vector prediction is computed by a fully connected layer, which maps the processed features to five outputs, corresponding to the three origin coordinates and two angular components of the gaze vector. This layer is followed by a layer of Parametric Leaky-Integrate-and-Fire (PLIF) neurons with their reset mechanism disabled, meaning they do not fire or reset their membrane potentials. Since gaze tracking is a regression problem, the membrane potentials of this final PLIF layer are directly interpreted as the output gaze vector.

The combination of convolutional and recurrent blocks in this architecture, containing a total of 2.3M parameters, enables the network to effectively leverage both the spatial structure and temporal dynamics inherent in the input event streams. This synergy allows GazeSCRNN to process event-based data effectively, making it particularly well-suited for gaze tracking applications.

#### D. Metrics and Loss

For gaze tracking, we predict gaze vectors. These vectors consist of an origin point, which consists of three coordinates in mm, and two spherical coordinates, representing the direction where the gaze vector is pointing. Therefore, to measure the accuracy of the predicted gaze vectors, we compute two important metrics. We compute the distance between the predicted origin point and the ground-truth pupil coordinates from the target data, which we will call the pupil error. Furthermore, we compute the angle between the predicted gaze vector and the ground truth gaze vector from the target data, which we will call the angle error. The accuracy of each model can thus be summarized by the Mean Pupil Error (MPE) and the Mean Angle Error (MAE).

To optimize our model, we use a loss function that minimizes these two errors. For this, we use a combination of a Mean Squared Error Loss for the pupil distance, and a Cosine Distance loss for the angles. For our experiments, we have chosen to select our best models based on their Mean Angle Error, because this is the metric that currently needs the most improvement.

In addition to the MPE and MAE metrics, we also measure the Mean Firing Rate (MFR) of the spiking neurons, which represents the mean ratio of neurons that spike during each time step, giving an indication of the sparsity of the spikes in the network. More sparsity, meaning fewer spikes, can enhance processing efficiency, particularly on neuromorphic hardware. However, it is important to acknowledge that sparsity is not currently a primary optimization goal. Although, It is possible

to also optimize for sparsity by including it as an additional term in the loss.

#### E. Training

All our models are trained using backpropagation and stochastic gradient descent. Since the spikes generated by our spiking neurons are discontinuous and therefore non-differentiable, we employ the surrogate gradient method to compute the gradients for these spiking neurons [9].

As described in Section III-C, our model is time dependent, meaning that computation at each time step depends on prior time steps. Training models with such temporal dependencies often relies on Backpropagation-Through-Time (BPTT), which unrolls the entire network across all time steps to propagate the loss backward through the sequence. However, BPTT incurs substantial memory costs that grow with the input sequence length, making it impractical for long sequences, as in our case.

To address these challenges, we use two methods to manage memory and computational complexity: truncated Backpropagation-Through-Time and Forward-Propagation-Through-Time (FPTT). In truncated BPTT, the loss is propagated back only to the inputs from the last few time steps, rather than the entire sequence. By selecting an optimal truncation length, we can balance memory consumption and computational costs, with a high enough backpropagation depth for good accuracy.

In addition to truncated BPTT, we apply Forward-Propagation-Through-Time (FPTT) [11], [21], which offers a significant advantage for training adaptive spiking neural networks with long temporal dependencies. FPTT minimizes the computation and memory requirements associated with BPTT by computing updates based on an instantaneous loss function. This loss is adjusted with a dynamic regularization term based on a running average of the model parameters, which stabilizes the gradients without accumulating error from previous steps, as is often seen with BPTT in recurrent neural networks. This approach removes the need to unroll the network across all time steps, allowing FPTT to train on long sequences with fixed memory requirements [11], [21]. Since we are dealing with a sequence-to-sequence problem, computing the instantaneous loss at every time step is trivial. Furthermore, this immediate, time-localized updating not only minimizes memory usage but also supports online training, which can be beneficial for real-time applications such as gaze tracking, for example to allow for personal calibration by the end user.

Yin et al. in [11] demonstrated that FPTT achieves improved accuracy and memory efficiency when paired with Adaptive Leaky-Integrate-and-Fire (ALIF) neurons with Liquid Time Constants (LTCs). In our model, LTCs allow neurons to dynamically adjust their behaviour, such as membrane potential decay and threshold adaptation, based on the inputs and prior neuronal states. This adaptability enables FPTT to optimize the spiking dynamics of the neurons, making it highly effective in capturing both short- and long-term dependencies with minimal computational overhead [11], which is advantageous for resource-constrained applications like wearable and edge devices [22].

In our experiments, we apply both FPTT and truncated BPTT, as well as a hybrid approach that combines FPTT with backpropagation over a limited number of time steps. In Section IV-A, we present an analysis of the effects of different backpropagation depths, as well as the benefits of FPTT for training our models efficiently and accurately.

#### IV. ABLATION STUDIES

This section presents ablation studies that assess the impact of specific features and hyperparameters on model accuracy (Mean Angle Error, MAE; Mean Pupil Error, MPE) and efficiency (Mean Firing Rate, MFR).

We use the GazeSCRNN architecture described in Section III-C. Each experiment modifies a particular feature or hyperparameter while training on the training dataset and evaluating performance per epoch on the validation dataset. The final evaluation uses a previously unseen test dataset, selecting the model with the best MAE on the validation set, as MAE is deemed the most significant metric for model usability and accuracy at this time.

##### A. Experiment 1: Effect of FPTT and BPTT Time Steps

We investigate Forward-Propagation-Through-Time (FPTT) and variations in the number of truncated Backpropagation-Through-Time (BPTT) steps, denoted as  $T$ . In our implementation, training applies BPTT to the last  $T$  time steps after waiting for  $T$  steps during forward propagation. Increasing  $T$  reduces the optimizer updates per epoch, as each input frame (representing one time step) propagates through the model only once. Results, shown in Table I and Figure 4, reveal that both FPTT and an optimal  $T$  improve the accuracy of the model. Applying FPTT shows an improvement in accuracy, especially when  $T$  is lower. As  $T$  increases, the need for FPTT diminishes. For subsequent experiments, we will use FPTT with  $T = 8$  to train the models.

##### B. Experiment 2: Event Framing Method

We assess the effects of converting event streams into frames using two methods: fixed event counts per frame and fixed time windows per frame. Note that the framing method and granularity affect the total number of frames in the dataset. So, the test and validation sets need to be redrawn for each framing method, meaning the results from this experiment all use a different test subset. Table II and Figure 5 present results,

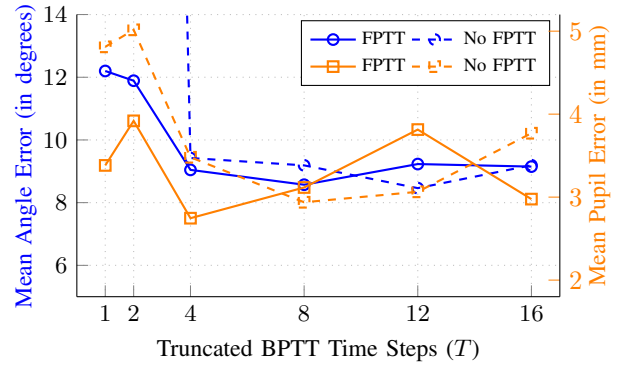


Fig. 4. Mean Angle Error (MAE) and Mean Pupil Error (MPE) vs. the number of Truncated Backpropagation-Through-Time Time Steps ( $T$ ) for models trained with and without Forward-Propagation-Through-Time (FPTT)

TABLE I  
ABLATION STUDY RESULTS OF APPLYING FPTT AND VARYING THE NUMBER OF TRUNCATED BPTT TIME STEPS FOR GAZESCRNN

| FPTT | BPTT Time Steps | MAE           | MPE     | MFR     |
|------|-----------------|---------------|---------|---------|
| ✓    | 1               | 12.20°        | 3.382mm | 0.03246 |
| ✓    | 2               | 11.89°        | 3.919mm | 0.03088 |
| ✓    | 4               | 9.045°        | 2.746mm | 0.02437 |
| ✓    | 8               | <b>8.572°</b> | 3.116mm | 0.02244 |
| ✓    | 12              | 9.231°        | 3.814mm | 0.03855 |
| ✓    | 16              | 9.150°        | 2.976mm | 0.02237 |
| ✗    | 1               | 86.25°        | 4.811mm | 0.03118 |
| ✗    | 2               | 86.26°        | 5.016mm | 0.02432 |
| ✗    | 4               | 9.424°        | 3.480mm | 0.02416 |
| ✗    | 8               | 9.189°        | 2.938mm | 0.03492 |
| ✗    | 12              | <b>8.463°</b> | 3.064mm | 0.03457 |
| ✗    | 16              | 9.193°        | 3.770mm | 0.03592 |

indicating that framing method and granularity significantly influence accuracy, with a fixed event count of 1750 achieving the best accuracy ( $MAE = 6.588^\circ$ ,  $MPE = 2.364\text{mm}$ ). Fixed time windows perform worse overall, with the optimal accuracy ( $MAE = 8.242^\circ$ ,  $MPE = 3.093\text{mm}$ ) achieved at 10ms, exactly matching the update rate for the gaze references from the Tobii Pro Glasses 3 at 100Hz, meaning the least interpolation.

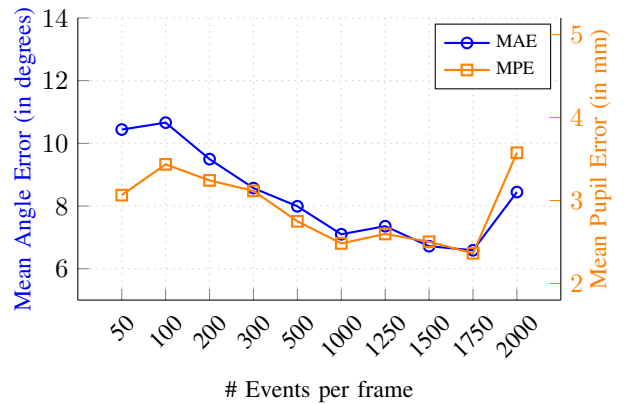


Fig. 5. Mean Angle Error (MAE) and Mean Pupil Error (MPE) vs. the number of events per aggregated frame in the input sequence

TABLE II  
ABLATION STUDY RESULTS OF VARYING FRAMING METHOD FOR GAZESCRNN

| Method      | # Events / Length | MAE           | MPE     | MFR     |
|-------------|-------------------|---------------|---------|---------|
| Event count | 50 events         | 10.44°        | 3.065mm | 0.01748 |
| Event count | 100 events        | 10.66°        | 3.436mm | 0.04163 |
| Event count | 200 events        | 9.496°        | 3.242mm | 0.02253 |
| Event count | 300 events        | 8.572°        | 3.116mm | 0.02244 |
| Event count | 500 events        | 7.992°        | 2.749mm | 0.02369 |
| Event count | 1000 events       | 7.100°        | 2.483mm | 0.04289 |
| Event count | 1250 events       | 7.357°        | 2.597mm | 0.03287 |
| Event count | 1500 events       | 6.719°        | 2.503mm | 0.03451 |
| Event count | 1750 events       | <b>6.588°</b> | 2.364mm | 0.04504 |
| Event count | 2000 events       | 8.446°        | 3.575mm | 0.03579 |
| Time window | 1ms               | 11.03°        | 3.828mm | 0.01348 |
| Time window | 5ms               | 9.575°        | 3.934mm | 0.01849 |
| Time window | 10ms              | 8.242°        | 3.093mm | 0.01855 |
| Time window | 20ms              | 13.33°        | 4.805mm | 0.02532 |

TABLE III  
ABLATION STUDY RESULTS OF USING DIFFERENT SPIKING NEURONS FOR GAZESCRNN

| Neurons | # Events | Training      | MAE           | MPE     | MFR     |
|---------|----------|---------------|---------------|---------|---------|
| ALIF    | 300      | FPTT, $T = 8$ | 8.572°        | 3.116mm | 0.02244 |
| PLIF    | 300      | FPTT, $T = 8$ | 12.89°        | 4.218mm | 0.02919 |
| ALIF    | 300      | $T = 12$      | 8.463°        | 3.064mm | 0.03457 |
| PLIF    | 300      | $T = 12$      | 12.14°        | 3.759mm | 0.06570 |
| ALIF    | 1750     | FPTT, $T = 8$ | <b>6.588°</b> | 2.364mm | 0.04504 |
| PLIF    | 1750     | FPTT, $T = 8$ | 8.451°        | 3.252mm | 0.04007 |
| ALIF    | 1750     | $T = 12$      | 6.965°        | 2.740mm | 0.03508 |
| PLIF    | 1750     | $T = 12$      | 10.94°        | 4.081mm | 0.04404 |

### C. Experiment 3: Neuron Types

This experiment investigates the impact of neuron complexity on model performance by comparing Adaptive Leaky-Integrate-and-Fire (ALIF) neurons, as described in Section III-B, to the simpler Parametric Leaky-Integrate-and-Fire (PLIF) neurons. For this, all ALIF layers in the GazeSCRNN architecture were replaced with PLIF layers, resulting in a model with only 790k parameters, compared to the 2.3M parameters in the original architecture. Both configurations were evaluated using aggregated frames with 300 and 1750 events, chosen based on findings in Section IV-B, and were trained using either FPTT with 8 truncated BPTT time steps, or 12 truncated BPTT time steps without FPTT, based on findings in Section IV-A.

The results, shown in Table III, indicate that models using PLIF neurons are significantly less accurate than those with ALIF neurons in all tested configurations. While the reduction in parameters highlights the computational efficiency of the PLIF-based model, this simplification leads to significant increases in MAE and MPE. These findings underscore the importance of the adaptive mechanisms provided by ALIF neurons, which appear to capture temporal dependencies more effectively.

### D. Experiment 4: Data Augmentation

We evaluate the impact of augmenting input frames of 300 events per frame with random affine transformations (rotations, translations, rescaling, and shearing) and random noise injection (see Section III-A1). Table IV shows that the

TABLE IV  
ABLATION STUDY RESULTS OF APPLYING DATA AUGMENTATION TO INPUT FRAMES FOR GAZESCRNN

| Affine | Noise Probability | MAE           | MPE     | MFR     |
|--------|-------------------|---------------|---------|---------|
| ✗      | 0.00              | <b>8.572°</b> | 3.116mm | 0.02244 |
| ✓      | 0.00              | 8.595°        | 2.606mm | 0.04563 |
| ✗      | 0.005             | 9.588°        | 3.126mm | 0.05685 |
| ✗      | 0.01              | 10.56°        | 3.277mm | 0.05828 |
| ✗      | 0.05              | 13.53°        | 4.325mm | 0.08703 |

TABLE V  
ABLATION STUDY RESULTS OF EXCLUDING UNRELIABLE INTERPOLATED REFERENCES FROM LOSS AND METRICS FOR GAZESCRNN

| Threshold | # Events    | Included | MAE           | MPE     | MFR     |
|-----------|-------------|----------|---------------|---------|---------|
| N/A       | 300 events  | 100%     | 8.572°        | 3.116mm | 0.02244 |
| N/A       | 1500 events | 100%     | 6.719°        | 2.503mm | 0.03451 |
| N/A       | 1750 events | 100%     | 6.588°        | 2.364mm | 0.04504 |
| 50ms      | 300 events  | 84.98%   | 8.396°        | 2.829mm | 0.02337 |
| 50ms      | 1500 events | 84.98%   | 6.156°        | 2.199mm | 0.03263 |
| 50ms      | 1750 events | 84.99%   | 6.56°         | 2.482mm | 0.05066 |
| 20ms      | 300 events  | 59.87%   | 8.807°        | 2.650mm | 0.02498 |
| 20ms      | 1500 events | 59.88%   | <b>6.034°</b> | 2.094mm | 0.03391 |
| 20ms      | 1750 events | 59.89%   | 6.216°        | 2.369mm | 0.07050 |

proposed data augmentation does not improve the accuracy of our model. In our experiment, affine transformations do improve MPE, but it is important to note that our best models are chosen solely on the MAE, not the MPE. Injecting random noise increases MAE significantly, suggesting that noise disrupts critical event patterns needed for accurate predictions. It is also important to note that the data augmentations are randomized for each epoch, increasing training time.

### E. Experiment 5: Excluding Unreliable Interpolated References

This experiment examines the exclusion of unreliable interpolated gaze references from loss and metric computations. Unreliable references arise when long intervals between ground truth gaze vectors from the Tobii Pro Glasses 3 require extensive interpolation. We define thresholds for acceptable interpolation durations, masking references exceeding these thresholds while retaining their sequence data for membrane potential and recurrent connection dynamics. Results, summarized in Table V, demonstrate that excluding unreliable references can improve MAE and MPE, particularly when paired with larger event counts per frame and tighter thresholds. However, this is not necessarily the case, as can be seen in the result with 300 events per frame and a maximum threshold of 20ms, which has worse accuracy.

### F. Experiment 6: Using Full Size Inputs

In the default setup, all models are trained using input frames downsampled by a factor of two, resulting in dimensions of  $173 \times 130 \times 2$ . For this experiment, we evaluate the impact of training the model on full-size frames with dimensions of  $346 \times 260 \times 2$ , containing either 300 or 1750 events per frame. To accommodate the increased input size, an additional convolutional block is introduced, as detailed in Section III-C. The results, shown in Table VI, indicate that using full-size

TABLE VI  
ABLATION STUDY RESULTS OF USING FULL SIZE INPUTS FOR  
GAZE SCRNN

| Full Size Input | # Events    | MAE           | MPE     | MFR     |
|-----------------|-------------|---------------|---------|---------|
| ✗               | 300 events  | 8.572°        | 3.116mm | 0.02244 |
| ✓               | 300 events  | 8.396°        | 2.676mm | 0.01083 |
| ✗               | 1750 events | <b>6.588°</b> | 2.364mm | 0.04504 |
| ✓               | 1750 events | 8.355°        | 3.351mm | 0.07456 |

inputs does not significantly improve accuracy for an event count of 300 events per frame. Furthermore, for an event count of 1750 events per frame, using full-size inputs significantly decreases accuracy.

## V. DISCUSSION

The GazeSCRNN model highlights the potential of spiking neural networks (SNNs) for event-based gaze tracking, but also reveals several limitations that suggest areas for improvement. A significant challenge lies in the reliance on gaze reference data generated by the Tobii Pro Glasses 3 gaze tracker. We observed prolonged periods where it failed to provide valid gaze vectors, leading to interpolation between widely separated ground truth values, giving unreliable targets. Excluding unreliable interpolated references above a time window threshold improves model accuracy but reduces the amount of usable training data, particularly when tight thresholds are applied. This trade-off underscores a broader limitation in the quality of ground truth data and suggests a need for alternative strategies for generating robust ground truth targets.

The preprocessing method for framing event streams also presents important trade-offs. Two framing methods were examined in this paper: fixed event counts and fixed time windows. Fixed event count framing results in a dynamic frame rate that adapts to the rate of incoming events from the Dynamic Vision Sensor (DVS) camera, aligning with the asynchronous nature of the DVS camera, by only providing updates when there are observed eye movements. This allows the model to focus on meaningful changes in the user’s gaze, rather than processing redundant information. However, this approach means that each frame represents a variable timescale, which can complicate the interpretation of temporal information. Despite this, fixed event count framing demonstrated superior accuracy compared to fixed time window framing. This is likely because fixed event counts ensure that every frame contains sufficient information to make accurate predictions, while fixed time windows may yield frames with highly variable and often insufficient event counts.

Another consideration is the relationship between framing choices and the gaze tracker’s update rate. The number of events per frame or the duration of the time windows determine the frame rate of the input sequence, which in turn can affect the output update rate of the gaze tracker. For example, if the system requires aggregation of frames before making predictions, the gaze tracker’s update rate becomes tied to the input frame rate, potentially introducing latency. Future research could explore alternative methods, such as rolling

windows or asynchronous spike processing, to address these challenges and provide more frequent predictions.

The computational demands of the GazeSCRNN model represent another area for improvement. Although the architecture is relatively simple, it relies on the complex and trainable spiking behaviour of the Adaptive Leaky-Integrate-and-Fire (ALIF) neurons to provide versatility and accuracy. While ALIF neurons are integral to capturing temporal dependencies effectively, they significantly increase the parameter count compared to simpler Parametric Leaky-Integrate-and-Fire (PLIF) neurons. This added complexity poses challenges for deploying the model on resource-constrained hardware. Future work could investigate hybrid neuron architectures or alternative adaptive mechanisms that maintain temporal modelling capabilities of ALIF neurons while reducing computational overhead. Moreover, emerging architectures such as spiking transformers [23], [24] and spiking state space models [25] could be explored for event-based gaze tracking, potentially offering novel ways to improve both accuracy and efficiency.

To unleash the full potential of the speed and energy efficiency of event-based gaze tracking, the GazeSCRNN model would ideally be implemented on neuromorphic hardware optimized for spiking neural networks. Such hardware could leverage the sparsity of spiking computations, enhancing both energy efficiency and processing speed [7]. We recommend this as a focus for further research, evaluating the performance of GazeSCRNN in terms of latency and energy consumption, and benchmarking it against existing gaze tracking systems. This would provide critical insights into the practical applicability of our work in real-world settings.

## VI. CONCLUSION

This paper introduces GazeSCRNN, a spiking convolutional recurrent neural network designed for event-based gaze tracking, leveraging the high temporal resolution and asynchronous nature of Dynamic Vision Sensor (DVS) cameras.

This work demonstrates the viability and potential of SNNs for event-based gaze tracking, while identifying key challenges and opportunities for improvement. By addressing these limitations, the GazeSCRNN model can contribute to advancing the field of gaze tracking, particularly for applications requiring high temporal precision and energy efficiency. The findings of this work pave the way for future research endeavours, such as enhancing the quality of ground truth data, exploring alternative framing strategies such as rolling windows or asynchronous spike processing, and implementing the model on neuromorphic hardware. Furthermore, advancements in spiking neural network architectures, including spiking transformers and spiking state space models, offer promising directions for improving both efficiency and accuracy.

## REFERENCES

- [1] D. W. Hansen and Q. Ji, “In the Eye of the Beholder: A Survey of Models for Eyes and Gaze,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 478–500, Mar. 2010.

- [2] J. Liu, J. Chi, H. Yang, and X. Yin, "In the eye of the beholder: A survey of gaze tracking techniques," *Pattern Recognition*, vol. 132, p. 108944, Dec. 2022.
- [3] A. T. Duchowski, *Eye Tracking Methodology*. Cham: Springer International Publishing, 2017.
- [4] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-Based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, Jan. 2022.
- [5] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 Ms Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.
- [6] M. Shahsavari, D. Thomas, M. van Gerven, A. Brown, and W. Luk, "Advancements in spiking neural network communication and synchronization techniques for event-driven neuromorphic systems," *Array*, vol. 20, p. 100323, Dec. 2023.
- [7] M. Pfeiffer and T. Pfeil, "Deep Learning With Spiking Neurons: Opportunities and Challenges," *Frontiers in Neuroscience*, vol. 12, Oct. 2018.
- [8] G. Zhao, Y. Yang, J. Liu, N. Chen, Y. Shen, H. Wen, and G. Lan, "EV-Eye : Rethinking high-frequency eye tracking through the lenses of event cameras," in *Thirty-Seventh Conference on Neural Information Processing Systems (NeurIPS), 2023*, New Orleans, USA, Sep. 2023.
- [9] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, Nov. 2019.
- [10] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and Learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.
- [11] B. Yin, F. Corradi, and S. M. Bohte, "Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time," Nov. 2022.
- [12] J. Jiang, X. Zhou, S. Chan, and S. Chen, "Appearance-Based Gaze Tracking: A Brief Review," in *Intelligent Robotics and Applications*, H. Yu, J. Liu, L. Liu, Z. Ju, Y. Liu, and D. Zhou, Eds. Cham: Springer International Publishing, 2019, pp. 629–640.
- [13] A. N. Angelopoulos, J. N. Martel, A. P. Kohli, J. Conradt, and G. Wetzstein, "Event-Based Near-Eye Gaze Tracking Beyond 10,000 Hz," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2577–2586, May 2021.
- [14] C. Ryan, B. O'Sullivan, A. Elrasad, A. Cahill, J. Lemley, P. KIELTY, C. Posch, and E. Perot, "Real-time face & eye tracking and blink detection using event cameras," *Neural Networks*, vol. 141, pp. 87–97, Sep. 2021.
- [15] Q. Chen, Z. Wang, S.-C. Liu, and C. Gao, "3ET: Efficient Event-based Eye Tracking using a Change-Based ConvLSTM Network," in *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2023, pp. 1–5.
- [16] Y. Jiang, W. Wang, L. Yu, and C. He, "Eye Tracking Based on Event Camera and Spiking Neural Network," *Electronics*, vol. 13, no. 14, p. 2879, Jan. 2024.
- [17] P. Bonazzi, S. Bian, G. Lippolis, Y. Li, S. Sheik, and M. Magno, "Retina : Low-Power Eye Tracking with Event Camera and Spiking Hardware," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5684–5692.
- [18] iniVation AG, "DAVIS346 Documentation," Nov. 2024.
- [19] Tobii AB, "Tobii Pro Glasses 3 Product Description," Jun. 2024.
- [20] B. Yin, F. Corradi, and S. M. Bohte, "Effective and Efficient Computation with Multiple-timescale Spiking Recurrent Neural Networks," in *International Conference on Neuromorphic Systems 2020*, ser. ICONS 2020. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 1–8.
- [21] A. Kag and V. Saligrama, "Training Recurrent Neural Networks via Forward Propagation Through Time," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 5189–5200.
- [22] M. H. Varposhti, M. Shahsavari, and M. van Gerven, "Energy-Efficient Spiking Recurrent Neural Network for Gesture Recognition on Embedded GPUs," Aug. 2024.
- [23] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. Yan, Y. Tian, and L. Yuan, "Spikformer: When Spiking Neural Network Meets Transformer," in *The Eleventh International Conference on Learning Representations*, Sep. 2022.
- [24] M. Yao, J. Hu, Z. Zhou, L. Yuan, Y. Tian, B. Xu, and G. Li, "Spike-driven Transformer," *Advances in Neural Information Processing Systems*, vol. 36, pp. 64 043–64 058, Dec. 2023.
- [25] W. Li, X. Hong, R. Xiong, and X. Fan, "SpikeMba: Multi-Modal Spiking Saliency Mamba for Temporal Video Grounding," May 2024.
- [26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017.
- [27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Jun. 2021.
- [28] Z. Zhou, K. Che, W. Fang, K. Tian, Y. Zhu, S. Yan, Y. Tian, and L. Yuan, "Spikformer V2: Join the High Accuracy Club on ImageNet with an SNN Ticket," Jan. 2024.
- [29] Y. Li, Y. Lei, and X. Yang, "Spikeformer: A Novel Architecture for Training High-Performance Low-Latency Spiking Neural Network," Nov. 2022.
- [30] —, "Spikeformer: Training high-performance spiking neural network with transformer," *Neurocomputing*, vol. 574, p. 127279, Mar. 2024.
- [31] M. Yao, J. Hu, T. Hu, Y. Xu, Z. Zhou, Y. Tian, B. Xu, and G. Li, "Spike-driven Transformer V2: Meta Spiking Neural Network Architecture Inspiring the Design of Next-generation Neuromorphic Chips," <https://arxiv.org/abs/2404.03663v1>, Feb. 2024.
- [32] R.-J. Zhu, Q. Zhao, G. Li, and J. Eshraghian, "SpikeGPT: Generative Pre-trained Language Model with Spiking Neural Networks," *Transactions on Machine Learning Research*, Apr. 2024.
- [33] Y. Wang, K. Shi, C. Lu, Y. Liu, M. Zhang, and H. Qu, "Spatial-Temporal Self-Attention for Asynchronous Spiking Neural Networks," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. Macau, SAR China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2023, pp. 3085–3093.

## APPENDIX

### SPIKING TRANSFORMER FOR GAZE TRACKING

In addition to the GazeSCRNN architecture proposed in this paper, we also experimented with a spiking transformer architecture for event-based gaze tracking. In this appendix, exclusive to the thesis, we will describe the architecture and results of this spiking transformer.

#### A. Related work

Spiking transformers are a relatively new way of designing spiking neural networks, based on the promising results of transformer models in various applications of deep learning [26], [27]. Transformer models use attention mechanisms to model relationships within the data. There are several spiking implementations of a transformer model, all of which have been introduced over the last few years [23], [24], [28]–[32]. One of the first proposed spiking transformer architectures is the so-called Spikformer [23]. The key difference between a vanilla transformer model and Spikformer is that the matrices used for the attention mechanisms ('Query', 'Key', and 'Value') consist of only spikes (1 or 0), unlike a vanilla transformer which uses floating point values. This significantly simplifies the required computations. Since the spike matrices can only contain non-negative values, the softmax function from the vanilla transformer can be omitted. Furthermore, matrix multiplications between the spike matrices are sparse, and boil down to only AND-gates and summations since the matrices contain only ones and zeros [23], allowing for efficient computation.

## B. Handling The Time Domain

The architecture of our spiking transformer models draw heavily from preceding spiking transformer models. While spiking transformer architectures such as Spikformer [23] by Zhou et al., and Spike-driven Transformer (SDT) by Yao et al. [24], have demonstrated the effectiveness of spiking transformers for tasks like image and sequence classification, their designs are not inherently suited for sequence-to-sequence problems.

Transformer models process sequences of tokens, which in vision applications are typically used to represent spatial rather than temporal information [27]. Most existing spiking transformer architectures operate with time-independent layer operations, achieved by merging the time and batch dimensions during training and inference. While this approach works well for generating a single output at the end of a sequence, like image and sequence classification, it results in suboptimal performance for sequence-to-sequence tasks like gaze tracking, where predictions at each time step must incorporate preceding temporal context.

In this work, we examine two methods for incorporating temporal information into the transformer architecture. One approach, based on existing literature [33], leverages attention mechanisms on both the spatial and the temporal domain. We apply the Spatial-Temporal Spiking Transformer (STS-Transformer) architecture by Wang et al. [33] to our gaze tracking task, leveraging spatio-temporal attention mechanisms.

In addition, we examine a different approach in which we make the spiking transformer model, based on the Spike-driven Transformer architecture (SDT) by Yao et al. [24], explicitly time-dependent by adapting the spiking neuron layers with recurrent connections, similar to the design in the GazeSCRNN architecture. We will call this architecture GazeSRTransformer.

## C. Spatial-Temporal Spiking Transformer Architecture

We apply the Spatial-Temporal Spiking Transformer (STS-Transformer) proposed by Wang et al. [33] to the event-based gaze tracking task. The STS-Transformer leverages a Spatial-Temporal Self-Attention (STSA) mechanism to capture dependencies across spatial and temporal dimensions.

The architecture comprises an embedding stage, spiking transformer encoder blocks incorporating STSA and a multi-layer perceptron, and a regression head, resulting in parameter counts of 1.6M, 2.4M, and 4.0M, for depths of 1, 2, and 4 encoder blocks, respectively. The embedding stage processes the input event frames using a series of spiking convolutional layers, batch normalization, max-pooling, and Leaky Integrate-and-Fire (LIF) neurons. These layers progressively extract features and map them into a 256-dimensional spike embedding space.

At its core, the STS-Transformer architecture employs STSA within spiking transformer encoder blocks to model both spatial and temporal correlations. STSA operates by computing pairwise dependencies between tokens at different

spatial locations and time steps, using delayed synaptic interactions to mimic biological neurons [33]. A masking mechanism ensures causality by preventing information from future time steps from influencing earlier predictions. To further enhance its representation power, a Spatial-Temporal Relative Position Bias (STRPB) is integrated into the attention computation, embedding spatiotemporal position information into the token interactions [33].

The final gaze vector prediction is computed using a fully connected layer, which maps the processed embedding to five outputs corresponding to the three origin coordinates and two angular components of the gaze vector.

## D. GazeSRTransformer: a Spiking Recurrent Transformer Architecture

This architecture draws heavily from preceding spiking transformer models, most notably the Spike-driven Transformer (SDT) by Yao et al. [24]. However, instead of using time-independent layers, we make the GazeSRTransformer architecture explicitly time-dependent by adapting the spiking neuron layers with recurrent connections, similar to the design in the GazeSCRNN architecture. These recurrent connections allow the model to extract and retain information across time steps, making it well-suited for event-based sequence-to-sequence tasks like gaze tracking. Additionally, we replace standard spiking neurons with Adaptive Leaky-Integrate-and-Fire (ALIF) neurons, as described in Section III-B. This replacement facilitates more complex, adaptive, and trainable spiking behaviour, as well as increasing consistency between the GazeSRTransformer architecture and the GazeSCRNN architecture.

The GazeSRTransformer architecture consists of a patch embedding stage, spiking transformer blocks, and a regression head, resulting in a total of 3.6M parameters. The patch embedding stage employs a spiking patch splitting block composed of convolutional layers, batch normalization, max pooling layers, and ALIF neurons to process input event frames. This stage extracts features and maps them into 128-dimensional spike embedding space, leveraging recurrent connections and ALIF neurons to encode spatiotemporal features.

At the core of the model, the spiking transformer blocks include a spiking self-attention mechanism and a spiking multi-layer perceptron (MLP). The self-attention mechanism uses spiking neurons to compute the query, key, and value representations, which enhance embeddings by capturing spatial information at local and global scales. Recurrent connections within this mechanism ensure the integration of temporal dependencies. Similarly, the spiking MLP processes outputs from the self-attention mechanism and incorporates recurrent connections to reinforce temporal continuity.

As in the GazeSCRNN architecture, the final gaze vector prediction is computed using a fully connected layer, which maps the processed features to five outputs corresponding to the three origin coordinates and two angular components of the gaze vector. This is followed by a layer of Parametric Leaky-Integrate-and-Fire (PLIF) neurons with their reset mechanism

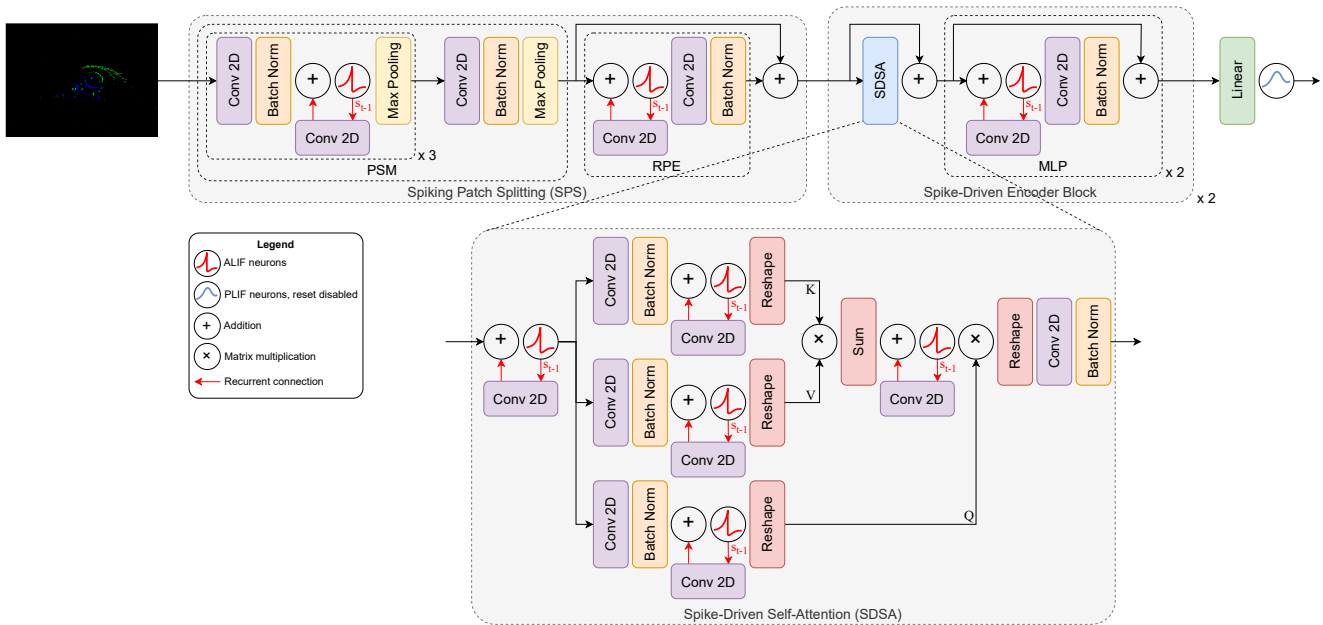


Fig. 6. GazeSRTransformer, a Spiking Transformer architecture for event-based gaze tracking

disabled, ensuring they do not fire or reset their membrane potentials. Since gaze tracking is a regression problem, the membrane potentials of the final PLIF layer are directly interpreted as the output gaze vector.

### E. Results

We evaluated both architectures on the preprocessed EV-Eye dataset, as described in Section III-A1. For this, the events were aggregated into frames of 1750 events each. The STS-Transformer architecture was trained with time steps of  $T = 10$  and  $T = 16$ , aligning with the configurations used in its original implementation [33]. In contrast, the GazeSRTransformer architecture, which processes each time step sequentially, was trained with a sequence of 1000 time steps, with a combination of Forward-Propagation-Through-Time (FPPT) and truncated Backpropagation-Through-Time (BPTT) for 8 time steps, consistent with the setup employed for the GazeSCRNN model, as described in Section III-E and Section IV-A.

Table VII summarizes the accuracy of the spiking transformer models. For the STS-Transformer, increasing the model depth has minimal impact on accuracy, but it results in a significant increase in model parameters. Furthermore, the STS-Transformer consistently underperforms compared to the GazeSRTransformer. The GazeSRTransformer achieves a substantially lower Mean Angular Error (MAE) of  $7.099^\circ$  compared to the STS-Transformer’s best result of  $10.46^\circ$ , demonstrating its superior capability in accurately predicting the gaze vector. Nevertheless, GazeSRTransformer is still outperformed by our GazeSCRNN model, which was able to achieve a Mean Angle Error (MAE) of  $6.034^\circ$  and a Mean Pupil Error (MPE) of  $2.094\text{mm}$ , using only  $2.3\text{M}$  parameters.

TABLE VII  
RESULTS FOR SPIKING TRANSFORMERS

| Architecture      | Time steps | # Parameters | MAE                             | MPE     |
|-------------------|------------|--------------|---------------------------------|---------|
| STS-Transformer-1 | $T = 10$   | 1.6M         | $10.67^\circ$                   | 3.265mm |
| STS-Transformer-1 | $T = 16$   | 1.6M         | $10.61^\circ$                   | 3.394mm |
| STS-Transformer-2 | $T = 10$   | 2.4M         | $10.49^\circ$                   | 3.190mm |
| STS-Transformer-4 | $T = 10$   | 4.0M         | $10.46^\circ$                   | 3.218mm |
| GazeSRTransformer | $T = 1000$ | 3.6M         | <b><math>7.099^\circ</math></b> | 3.528mm |

### F. Conclusion

This appendix presented an exploration of spiking transformer architectures applied to event-based gaze tracking. Two distinct approaches were evaluated: the Spatial-Temporal Spiking Transformer (STS-Transformer) and the GazeSR-Transformer. While both architectures demonstrated the feasibility of leveraging spiking transformers for this task, their accuracies varied significantly.

The GazeSRTransformer, which incorporates recurrent spiking neuron layers to explicitly model temporal dependencies, achieved a significantly improved MAE of  $7.099^\circ$  compared to the STS-Transformer model from existing literature [33]. However, it remains outperformed by the GazeSCRNN architecture, which achieved the best overall accuracy with a lower parameter count.

These findings highlight the potential of spiking transformers for event-based tasks. However, while transformers and spiking transformer architectures have shown great promise in the past, we were ultimately unable to utilize them for creating an accurate gaze tracking model capable of outperforming our proposed GazeSCRNN architecture.